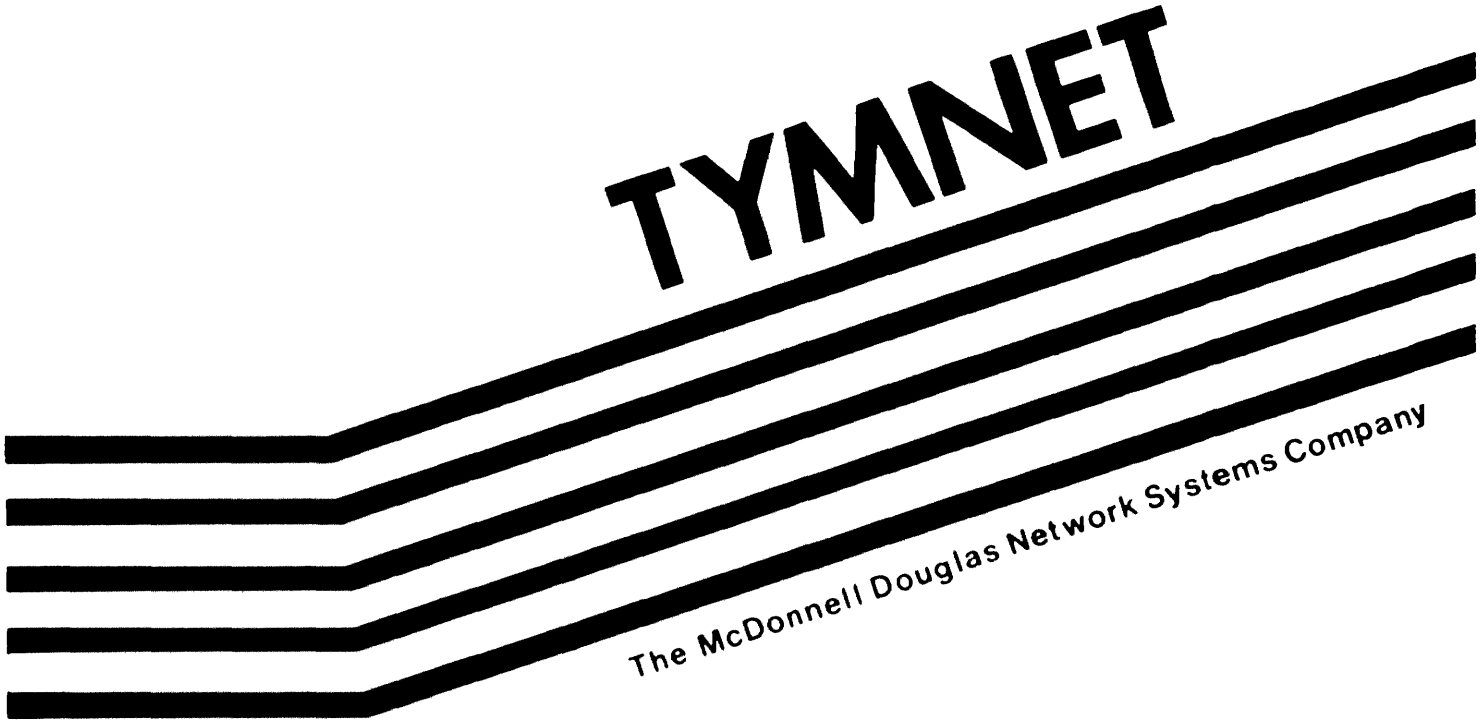
Network Products Concepts and Facilities



Network Products
Documentation

Network Products
Concepts and Facilities

January 15, 1985

Publication Number 57

FIGURES

TABLES

Hardware/Engine

CHAPTER 1

TYMNET COMMUNICATIONS PROCESSOR FAMILY

CONTENTS

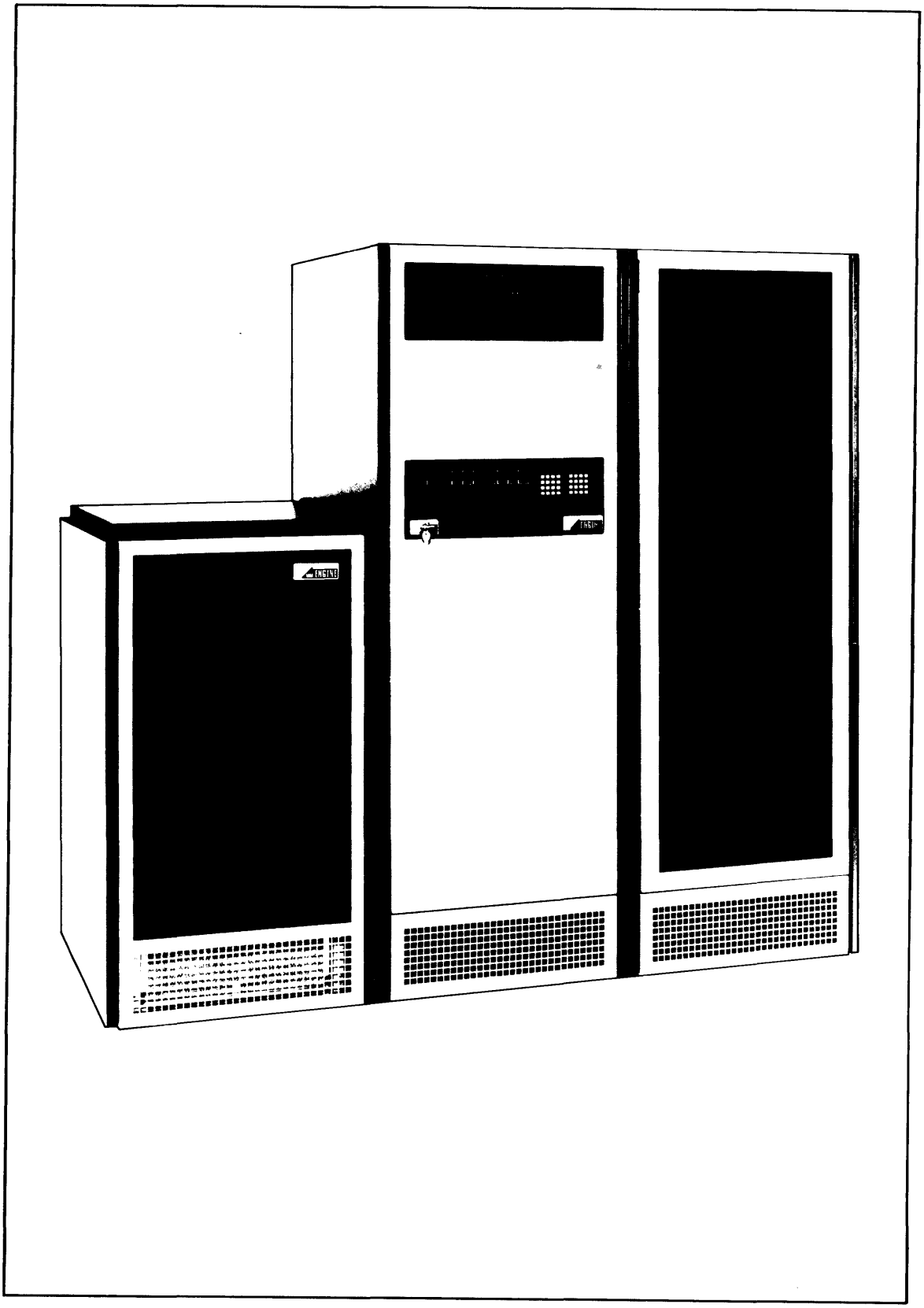


Figure 1-1. TYMNET Mini-Engine and TYMNET Engine Processor

OVERVIEW

Related Document:

- TYMNET Engine Maintenance Manual

TYMNET Communications Processor Family Product Line is as follows:

- TYMNET Engine Processor
- TYMNET Mini-Engine
- TYMNET Micro-Engine
- TYMNET Asynchronous Terminal Concentrator (ATC)
- Port Switch Module (PSM)
- Multiple Extended Processor (MP)
- Extended Processor Interface (XPI)
- TYMNET Super-Engine

PRODUCTS

TYMNET Engine Processor

TYMNET Engine Processors are high performance, microcode-driven, 32-bit minicomputers. Engine Processors operate as nodes in the TYMNET network with processor clusters used in applications where more capacity is needed.

The Engine has 16 sets of 16 general purpose registers, with internal micro-cycle timing of 125 nanoseconds. The processor has three Input/Output (I/O) interrupt levels and can contain up to 4 megabytes of semiconductor memory with special relocation and protection capabilities.

The physical description includes the following:

- cabinet
- blowers
- one 16-slot chassis
- power supply
- monitor panel
- systems console

In the 16-slot chassis, card placement for the first four slots are dedicated to the following:

- Central Processing Unit (CPU) board
- Read Only Memory (ROM) and I/O board
- Memory Access Controller (MAC) board
- Multifunction board

The other slots may be used for memory or I/O boards which do not need to interface to the Direct Memory Access (DMA) bus.

Power

The power supply can provide the following:

- input voltage: nominal 115 Vac
  47-63 Hz
  operating range 90-132 Vac

- output: outputs are isolated from each other and from the case to withstand 500 Vdc. The main output must be at least 10% loaded (total power rating) to obtain full load current from secondary outputs.

CH1	(main channel)	5.2V at 110A
CH2	(secondary channel)	16.5V at 6A
CH3	(secondary channel)	16.5V at 6A

Voltage adjustment is +/- 5% of nominal output (minimum adjustment range for all outputs). There is no need for internal adjustments.

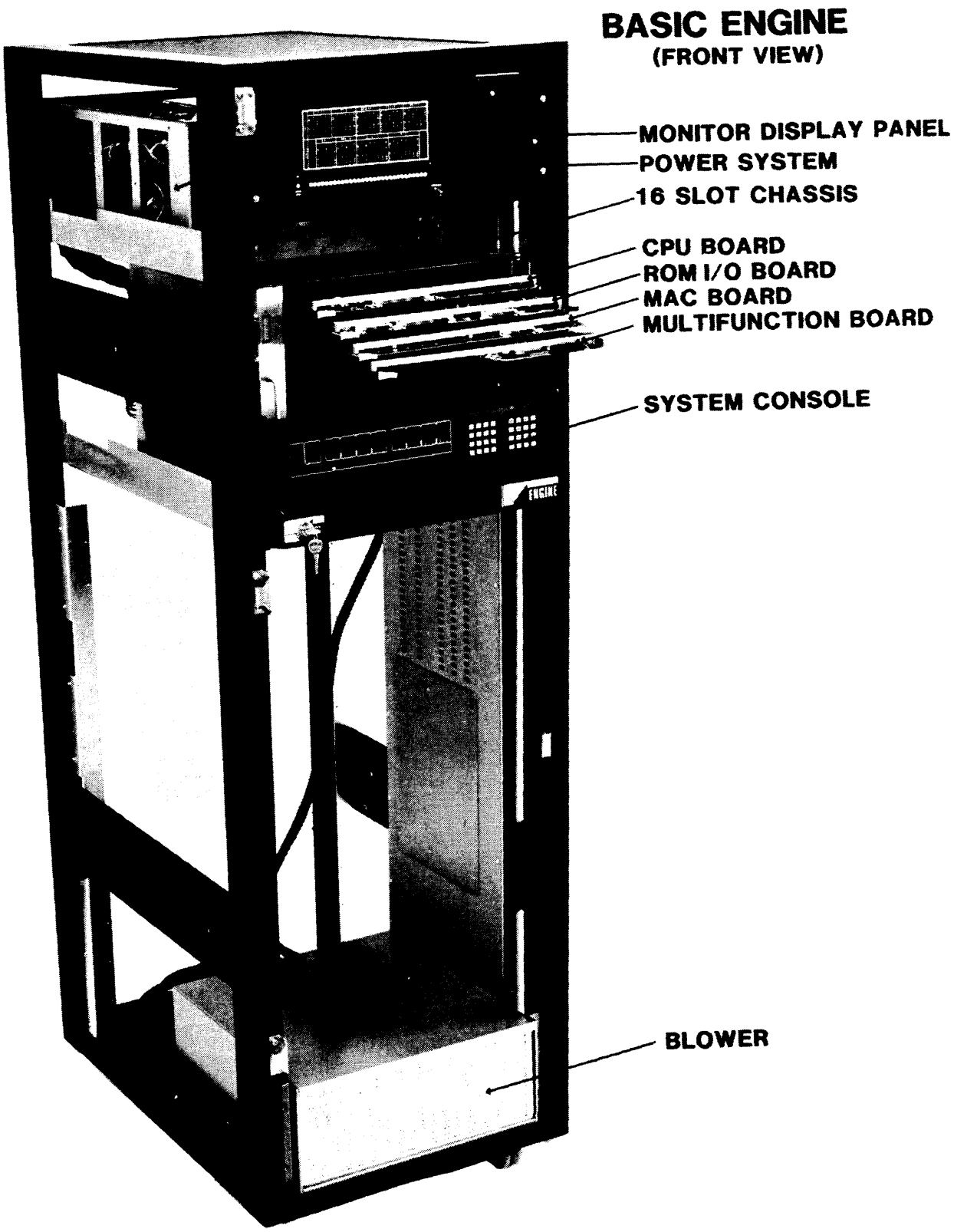


Figure 1-2. TYMNET Engine Minimum Configuration

Engine Minimum Configuration

Minimum requirements for machine operation are as follows:

- CPU board
- ROM and I/O board
- MAC board
- Multifunction board
- 1/4 megabyte of memory

Additional Boards

Three types of I/O boards:

- 32-port Asynchronous board, a maximum of eight boards can be installed in an Engine for up to 256 asynchronous lines.
- 16-port Synchronous board, a maximum of two synchronous boards can be installed in an Engine for up to 32 synchronous lines.
- Serial I/O (SIO) board, which is the DMA communications controller and handles up to eight dual-line interfaces. A maximum of four boards may be installed in an Engine.

The TYMNET Engine has a mass storage capability using 10, 160 or 300 Megabyte (Mb) disks and magnetic tapes. The actual throughput characteristics of the processor depend on the type and number of functions supported.

TYMNET Engine Applications

The Engine can function as a Supervisor node, switching node, a TYMCOM, or a TYMSAT, all operating under the Internally Switched Interface System (ISIS) control program.

ISIS is a special purpose operating system for communications processing. It allows several communication interface programs to operate simultaneously under multiprogramming. ISIS software consists of three major parts: the kernel (a control center), the dispatcher (a message carrier), and the slots (processes providing interface services). A slot is both a logical division and a software process. Interface boards and slots have no fixed one-to-one relationship. Interface boards are physical hardware interfaces, typically supporting a number of peripheral devices, such as terminals or printers. Each independent device can be separately assigned to a slot.

A board can be assigned to one or more slots. An asynchronous board, for example, can support 32 asynchronous ports which are assigned to slots in groups of 16. Thus, two slots could use a single asynchronous board. A synchronous interface board has 16 synchronous ports which could be divided among as many as 16 slots.

Alternatively, one slot can have several interface boards assigned to it. A single slot can support groups of asynchronous lines as well as synchronous lines.

In an Engine running under the ISIS control program, one slot may function as a gateway, another as a 3270 protocol, or another as a TYMSAT all under the same system. An Engine may be in a configuration in a private network, with an Automatic Port Switching System (APSS). These applications are just a few facilities that are supported by this product line.

TYMNET MINI-ENGINE

The TYMNET Mini-Engine is a smaller, lower priced version of the TYMNET Engine. It contains the same basic hardware as the Engine, but is mounted in a smaller cabinet, with an 8-slot chassis. The Mini-Engine has the same software and performance capabilities as the Engine and can operate the ISIS software environment.

The Mini-Engine supports SIO, disk, or tape devices, but does not support specialized network control functions, such as network supervision, or accounting collection.

It can be configured with up to three communications I/O boards, in various combinations of asynchronous, synchronous, SIO, or DMA devices.

Mini-Engine Applications

Some of the Mini-Engine applications are as follows:

- TYMSAT
- TYMCOM
- Switcher

The Mini-Engine has the same applications as a full size Engine would have. There also is a dual Mini-Engine which consists of two separate processors mounted in a tall cabinet.

TYMNET MICRO-ENGINE

The TYMNET Micro-Engine is the low-end version of the Engine family. It is a fully integrated system, packaged in a desk top enclosure consisting of three boards. They are the processor board, memory board and an I/O board functionally compatible with the basic TYMNET Engine. It also has add-on capability of a front console for easy maintenance.

The Micro-Engine is a 32-bit processor unit, with 16 asynchronous ports, and four synchronous ports using 256 or 512 kilobytes of semiconductor memory. With 512 Kb of semiconductor memory, the Micro-Engine will support up to eight synchronous ports.

Micro-Engine Applications

Some of the Micro-Engine applications are as follows:

- Remote asynchronous terminal interface
- Remote Packet Assembly/Disassembly (PAD) concentrator
- Asynchronous host interface
- IBM 3270 host or terminal interface
- IBM 2780/3780/HASP host or terminal interface
- Univac UTS-4000 interface
- Univac NTR interface

The Micro-Engine does not provide SIO ports. However, it can operate the ISIS software environment.

Asynchronous Terminal Concentrator (ATC)

The ATC is a compact communications processor providing access to a TYMNET packet-switched data network for up to eight remote or clustered asynchronous terminals. The ATC converts data from the terminals to TYMNET's synchronous protocol and concentrates it on one network line using multi-user packets for efficient transmission. The ATC has eight asynchronous terminal ports and two synchronous network ports. Based on a Very Large Scale Integration (VLSI) 16-bit microprocessor, the ATC is complete in a desk top cabinet with power supply, processor, memory and I/O interfaces. The ATC performs a hardware self-test when powered up and indicates operational status on the front panel.

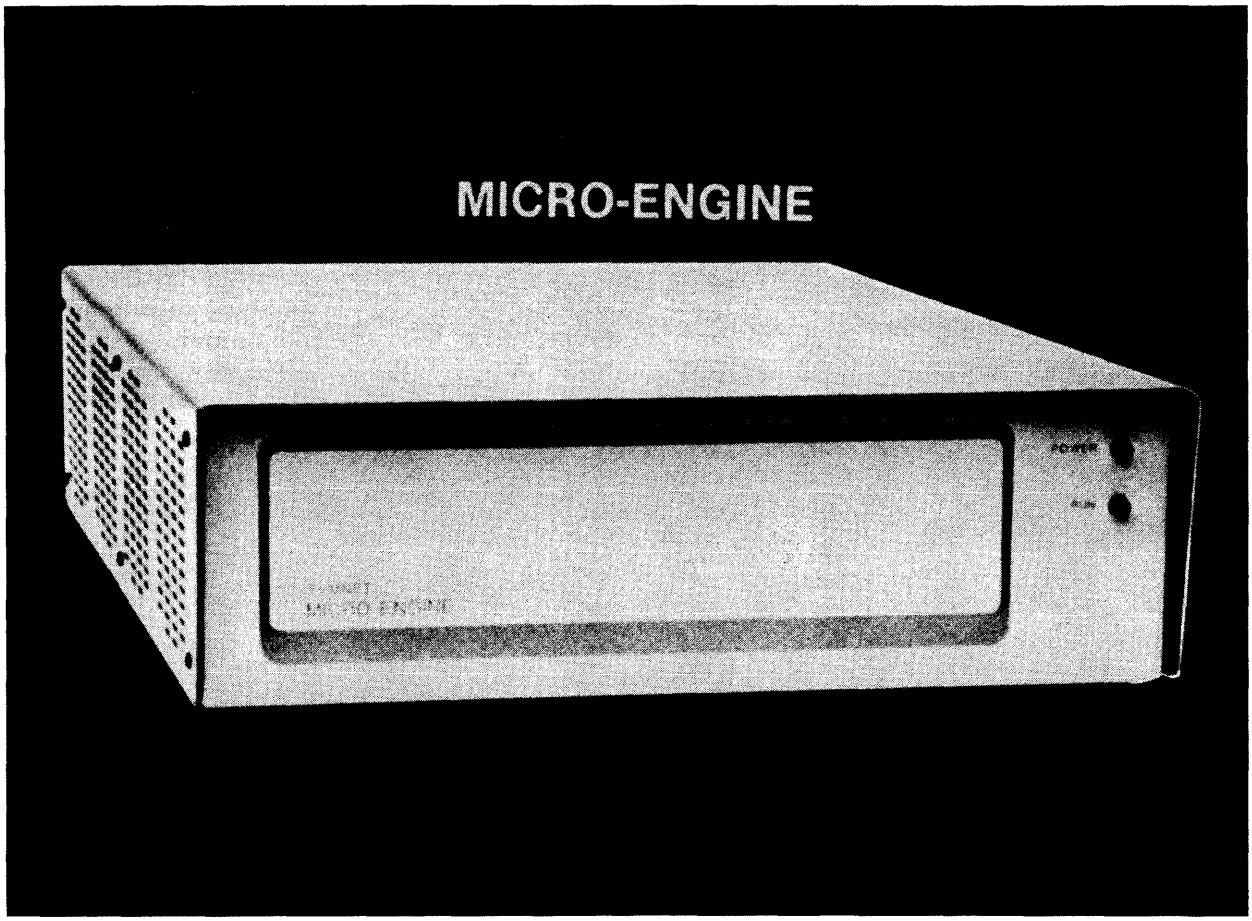


Figure 1-3. TYMNET Micro-Engine

ASSOCIATED SYSTEMS

Port Switch Module (PSM)

The PSM is the hardware for Automatic Port Switching System (APSS). It is a device that is used to switch network I/O ports from a failed machine to an operational one. Basically it is an electronic switch for I/O. The PSM is packaged in a card cage and contains a power supply module, control card module and various I/O interface modules. The PSM is mounted on isolated sliding racks inside the Engine cabinet.

The PSM consolidates three components of the current Engine system into one. It replaces the I/O connector panel, SIO daughter board card cage, and the front panel monitor assembly.

The PSM control card is plugged into the backplane connector of the card cage. The control card monitors information sent to it from the ISIS System Recovery Module (ISRM) slot in the Engine through what is called the HEARTBEAT. The HEARTBEAT is a 64-bit signal pattern that passes command and status information between the ISRM slot and the port switch module on a special dedicated non-SIO synchronous port.

Automatic switch control is determined by the HEARTBEAT criteria set in software and operator interface commands which are sent to the PSM by the HEARTBEAT signal. The manual override capability of automatic switching is provided by a switch on the front panel of the PSM control card. Manual override options are as follows:

- forced primary mode
- slave mode
- forced back-up mode
- normalize mode

LEDs on the front panel indicate the status of the switching function, PRIMARY or BACKUP.

ISIS Multiple Extended Processor (MP)

MP system moves the dispatcher capabilities from the Engine ISIS kernel to a DMA device called the Extended Processor Interface (XPI) board.

MP consists of a group of Engines located in the same facility that are interconnected using Ethernet protocol. These processors appear to the network as a single node, with enhanced capability. Each processor is configured with one megabyte of semiconductor memory, extended battery backup, and an XPI board. Port switching modules provide automatic switching of the communication lines if a failure should occur in one of the applications of the Engine.

MP provides increased network availability by using the Extended Processor Interface (XPI) to distribute application functions among resources in a node cluster. It also improves node capacity by allocating fewer functions to each processor.

Extended Processor Interface (XPI)

The XPI board is the hardware necessary for the MP concept. It is basically a DMA device with access to one megabyte of Engine memory. The board itself is based on Motorola's 68000 microprocessor. In the MP concept, the dispatcher capabilities are moved from the Engine ISIS kernel to the XPI board. The dispatcher function is now distributed among the other XPI boards in Engines in the cluster.

The XPI board has two serial ports: one for XPI communications, and another for downline loads directly into Engine memory. This board interfaces with an Ethernet Protocol module, which resides in the PSM.

The data flow for an MP system through an XPI board is as follows: a ribbon cable connection from the XPI board, which resides in an Engine, is connected to the Ethernet board in the PSM. From the Ethernet board, a cable in the PSM is attached to a Local Area Network (LAN) transceiver on one Engine, which allows communication to another LAN transceiver on another Engine in the cluster.

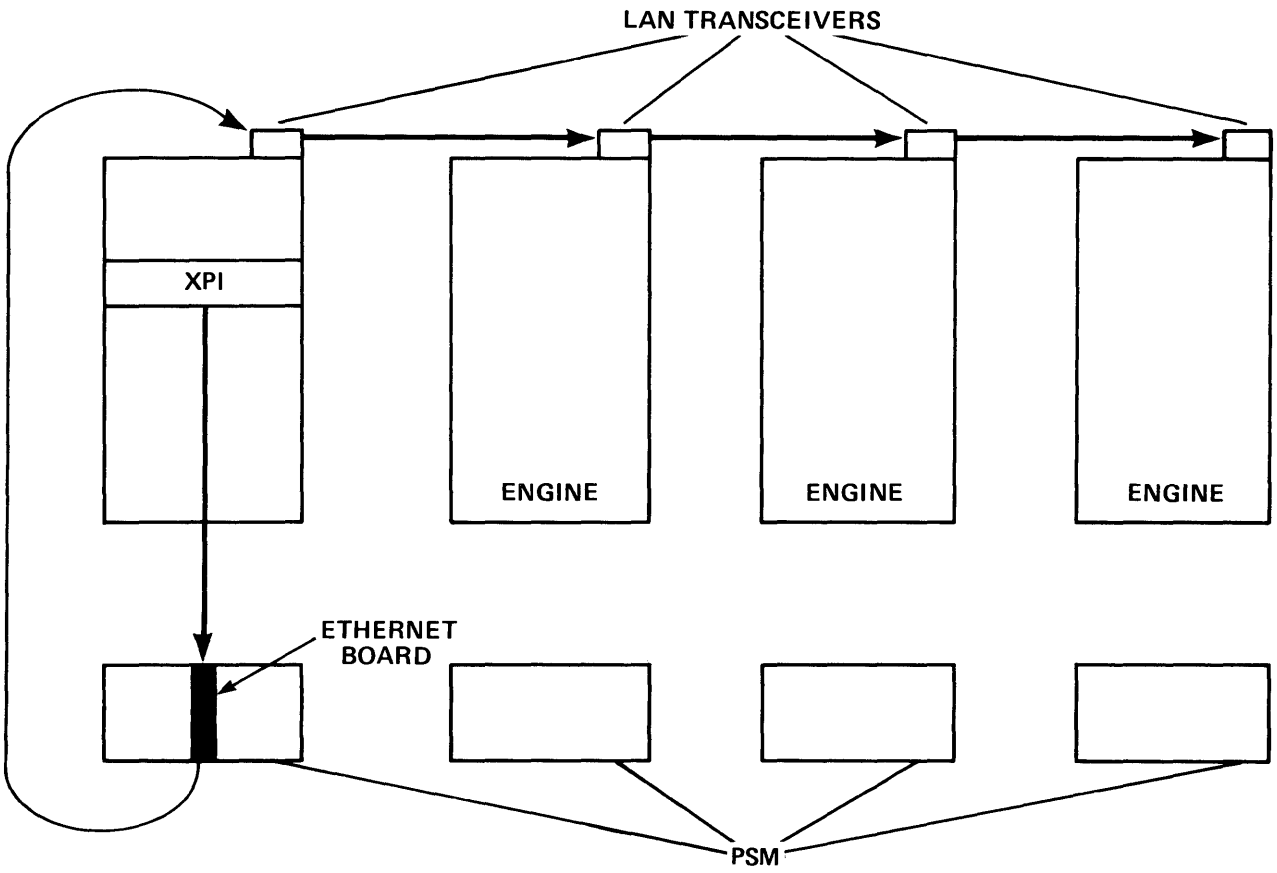


Figure 1-4. MP/XPI Systems Configuration

Super-Engine

The TYMNET Super-Engine is under development as the high-end version of the Engine family. It is designed for a wide range of data communications applications such as the following:

- high performance transit node
- wideband data communications networks

The Super-Engine's major features include the following:

- CPU - 10 times Engine CPU performance
- I/O - 10 to 100 times the Engine I/O throughput
- Memory - 4 to 32 Mb of physical memory

Two Super-Engine versions are planned, a phase one system with throughput of four megabits per second full duplex. The phase two system will have the throughput of 16 megabits/second full duplex.

The phase one system supports up to 64 lines at 64 kilobaud. The phase two system supports an I/O card cage that is configurable for disk or tape or communication lines with dedicated programmable processors.

The Super-Engine has a multiple processor architecture. The Main Processor (MP) executes instructions. The Channel Processor (CP) moves data between memory and communications interfaces. The Diagnostic Processor (DP) bootstraps the system, acts as a front panel, and provides system diagnostics.

CONTENTS

OVERVIEW

ISIS-II is TYMNET'S special purpose operating system for communications processing. It allows several communication interface programs to operate simultaneously under multiprogramming.

ISIS-II software consists of three major parts: the kernel (a control center), the dispatcher (a message carrier), and the slots (processes providing interface services).

ISIS-II functions like a miniature version of the network. Just as the Supervisor controls the whole network, ISIS-II controls the Engine and allocates the Engine's resources.

With ISIS-II, Engine memory is divided into slots that are logically separate partitions containing a collection of related processes. Up to 64 ISIS-II slots (job slots) may be allocated when an Engine is configured. Each interface board may be used by several slots. Each slot is independent, but all are controlled by the kernel, and may be interconnected by the dispatcher. This arrangement allows different interfaces and applications to reside in adjacent slots.

If slots communicate to an external device, for example, to a host computer using an interface board, they are termed "interfaces"; if the slots are entirely self-contained, they are termed "applications". Collectively, slot activities are termed "processes".

Figure 2-1 shows the general layout for an ISIS-II node configuration with three slots. The hardware device driver module services internal and external input and output (I/O) interrupts. The kernel schedules all other processes and provides communications between the hardware drivers and the individual slot processes. Each slot may have three job processes: Dynamic Debugging Tool (DDT), Foreground and Background, all of which may be active concurrently.

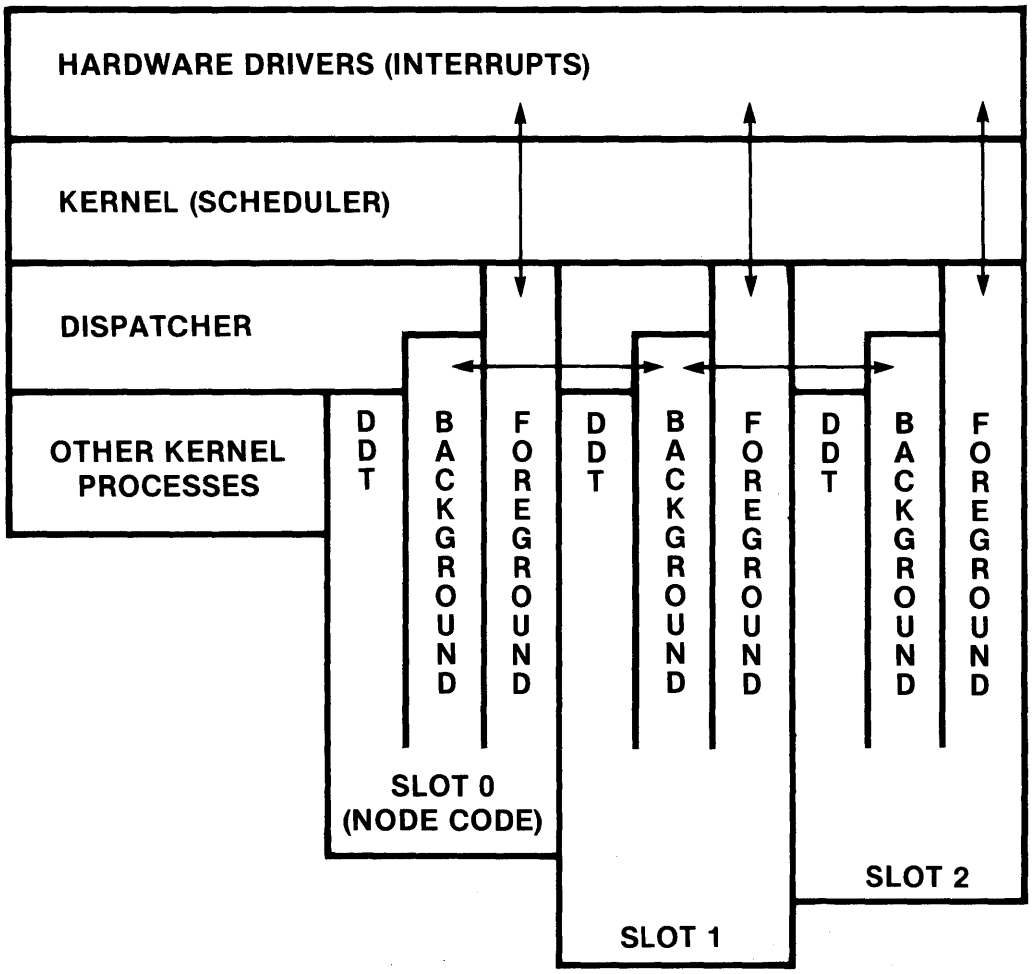


Figure 2-1. ISIS-II Schematic Arrangement

Slots are a concept unique to ISIS-II. To the ISIS-II operating system, the network interface is just another process in a slot. All ISIS-II need do is report periodically with accounting data to the network and make sure the slots operate properly. To the network, an ISIS-II node is just one of possibly hundreds of semi-independent units.

Node Code is a special interface program that contains the essential descriptive parameters of a node in the network and can perform the following.

- interfaces a node to the rest of the network
- creates virtual circuits
- handles data packet assembly and disassembly
- passes login requests to the Supervisor
- indicates link status to the Supervisor

With ISIS-II, Node Code always resides in slot 0. ISIS-II allows a node to perform routine network packet switching, virtual circuit building, and several other jobs, for example, synchronous and asynchronous interfacing. A node with ISIS-II Node Code can run as a TYMSAT, TYMCOM, use SIO links, and act as a network gateway, all on the same node.

All nodes are in contact with the network Supervisor through exclusive control channels in network links.

A typical Engine has the following standard boards: the Central Processing Unit (CPU) board, Read Only Memory (ROM) and I/O board, Memory Access Controller (MAC) board, Multifunction board, and a one megabyte Random Access Memory (RAM) board.

Additional interfaces often installed are asynchronous or bisynchronous V.25/RS-232-C, synchronous SDLC/HDLC, V.24 or V.35, and printer interface cards.

Figure 2-2 represents a close-up schematic view of a node in a small network.

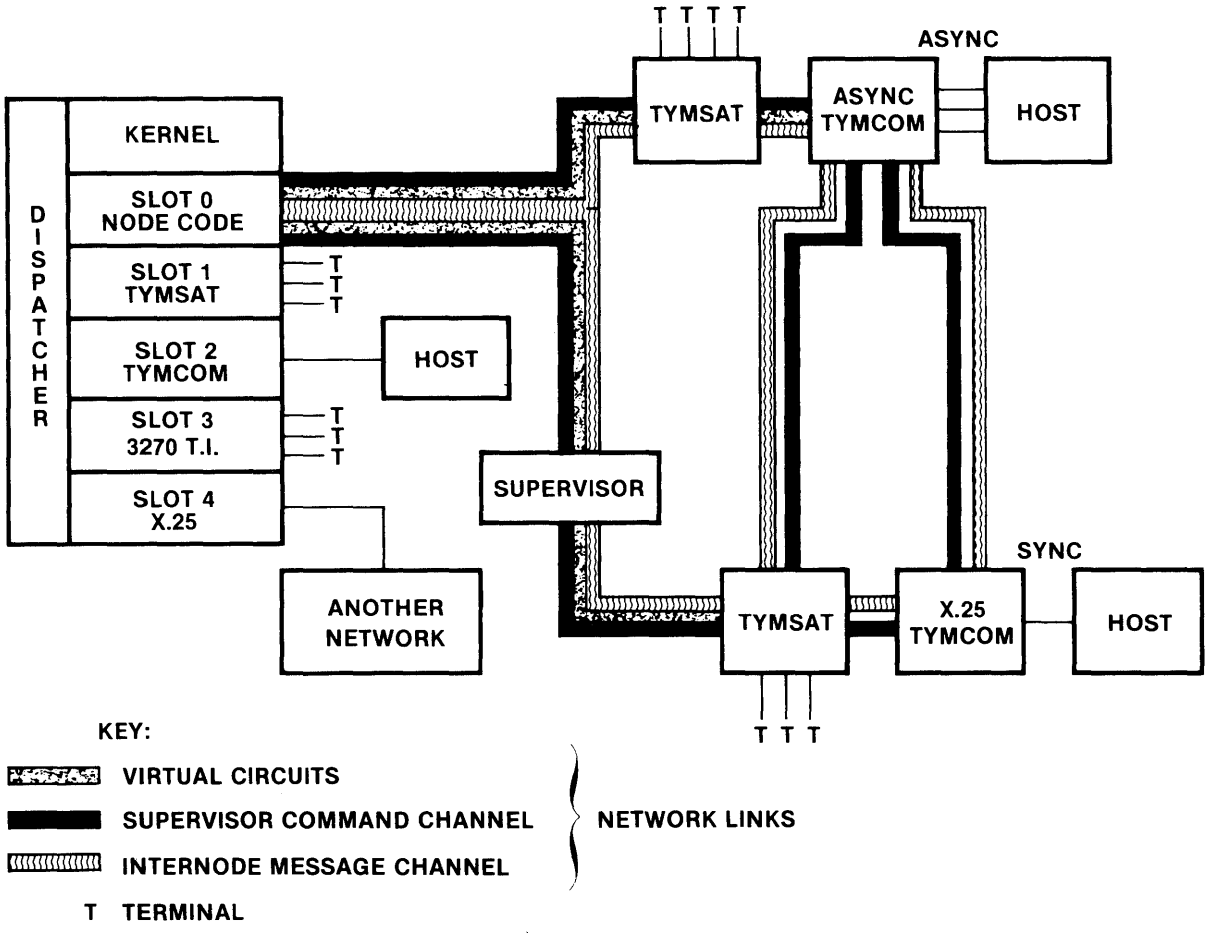


Figure 2-2. View of a Node in a TYMNET Network

The Kernel

The kernel is the control center of an Engine under ISIS-II and provides control over both Engine hardware and job scheduling. The kernel handles the following tasks.

- Manages job slots (interface and application processes) created under ISIS-II. Each interface runs one or more processes and is allocated its own area of memory, supervised by the MAC. Whatever Engine memory is not used by the kernel and dispatcher is available for the slots. The same functional process may run in different slots.

- Schedules CPU time for all processes, which it divides into jobs. Jobs are independently scheduled according to relative urgency. Interrupts have the highest priority followed by foreground, intermediate (dispatcher), and background jobs (including DDT for code debugging). Background jobs conduct the major part of a slot's work, receiving long periods of CPU time at nonuniform intervals. The kernel processes an interrupt and returns the CPU to the original job when the interrupt has been handled.

- Controls the MAC board that assigns physical memory addresses to logical slot process addresses. The kernel also updates segment F, a common read-only data storage area containing information on all current job processes in the slots.

- Handles software and hardware I/O drivers. ISIS-II employs a set of general purpose drivers to handle communications between hosts, terminals and peripheral devices. Centralized drivers provide a high level of process security and make interface process software more flexible.

- Processes Supervisor Calls (SVCs) from the job slots requesting service for functions which slots do not have the license to do. The kernel validates a slot's requests, servicing only those that are legitimate, thereby controlling total machine resources and maintaining system integrity.

The Dispatcher

The dispatcher is the communications switching center of ISIS-II, and handles all data switching between slots. To the dispatcher, all slots are seen as equal, except that Node Code (slot 0) is able to perform data packet routing and network accounting chores for the network Supervisor. The dispatcher collects and forwards this accounting data to Node Code during and at the end of job slot processing sessions.

The dispatcher runs as an intermediate level job that handles the ISIS-II internal data bus, linking the job slots with each other. The dispatcher operates each time a background job is run and sets up that background job for the next run. The dispatcher switches the output of a slot to the appropriate destination slot(s).

The slots must communicate with the dispatcher using a uniform format because all data from the interfaces in the slots must conform to a standard protocol. Each interface has a set of control tables used in communicating with the dispatcher and may also have a translator to convert messages to the standard message format. Each interface runs as one or more independent jobs under the kernel, sharing CPU time with other interfaces. The MAC prevents each job slot from destroying the memory of its neighbor slots by overwriting them. The dispatcher checks the data flow to ensure that an interface in error damages only its own subset of circuits.

The Slots

A slot is both a logical division and a software process. Interface boards and slots have no fixed one-to-one relationship. Interface boards are physical hardware interfaces, typically supporting a number of peripheral devices, such as terminals or printers. Each independent device can be separately assigned to a slot.

A board can be assigned to one or more slots. An asynchronous board, for example, can support 32 asynchronous ports which are assigned to slots in groups of 16. Thus, two slots could use a single asynchronous board. A synchronous interface board has 16 synchronous ports which could be divided among as many as 16 slots.

Alternatively, one slot can have several interface boards assigned to it. A single slot can support groups of asynchronous lines as well as synchronous lines.

Under MAC, a job slot can have up to 16 simultaneous memory segments, with hardware support for the segments provided by the MAC board. With ISIS-II, memory segment F is shared and readable by all processes. It contains system variables, such as clocks, which are of interest to all processes using CPU timeslices for job processing.

Segment E is a slot's Control Table Area (CTA) and is protected from inadvertent modification by the slot itself. The CTA contains descriptors of the slot's running configuration, such as pointers to variables, and memory size and layout. Each CTA is shared by the dispatcher and a slot using data structures to communicate with individual jobs and to record ongoing processes.

Dispatcher Internal Processes

ISIS-II internal communications are mainly between the dispatcher and the slots. When two interfaces talk to each other through the dispatcher, the sender and receiver do not have to be synchronous. What is necessary, however, is that the two data flows be independent of each other and that standard message format be used.

To accommodate bursts of data on the dispatcher bus, input and output ring buffers must be installed in all interface slots. Each ring buffer is allocated approximately a one second (maximum) data flow storage capacity.

Each interface, including slot 0, has a set of permuter tables that tell the dispatcher which port of one interface is connected to which port of another. Because slot processes are running concurrently, the dispatcher operates at an intermediate level of priority. This level is higher than background job priority, so background jobs get interrupted by the kernel's scheduler on behalf of the dispatcher.

The dispatcher normally attempts to move data from the source process to the destination process immediately. If the dispatcher cannot deliver at once, it buffers the data. To prevent source output ring buffer overloading, the dispatcher back-pressures the source port, not the source interface directly, as this would stop all source interface output, including output destined for other interfaces.

All external network I/O is routed through slot 0. The dispatcher, in turn, receives data from slot 0 and forwards it to the appropriate interface, constructing internal circuits as necessary.

If a slot requests a new circuit to a host, the dispatcher will notify and connect the slot 0 interface. When this circuit is made, the new circuit will be internal to the machine if the destination host is connected to the same machine. The dispatcher will do the necessary interconnection from source to destination process, so that data now flows entirely within the dispatcher without connecting with the slot 0 (Node Code) interface.

As the internal arrangement of data being transmitted between interfaces must be uniform, a standard message format is used. This format ensures that each message has a header, which specifies the destination port number and the message type. The message type supports a fixed length message and classifies transmitted data, eliminating much of the normal logic required to examine each character. Time is saved because the dispatcher need inspect only the message header to determine message routing.

ISIS-II External Communication

Slots do not communicate directly with external hosts, external terminals or other nodes in the network. When a slot sends data out of the node, it is routed through the kernel and slot 0. The kernel translates physical I/O devices into logical units used by the slots. This process is called centralized I/O control. Many peripheral devices, like printers, require a hardware driver (a short control program) to govern them. The kernel controls these drivers.

The majority of drivers are categorized into one of five communications classes: asynchronous, synchronous, SDLC/HDLC (bit-stuffing), channel, and particular I/O devices needing special drivers.

General purpose drivers may not be precisely suited to every application, but are adequate for most uses. The security value of centralized hardware drivers outweighs their relative lack of efficiency. The interprocess protection afforded, the ability to change or share devices without

rewriting each process using that device, and the lack of any specialized knowledge by the driver, makes process software easier to write and develop.

The use of logical units (one is assigned to each slot by the kernel) reduces the amount of code necessary in each slot, and, therefore, allows a slot to access only specified I/O devices. For example, a TYMSAT slot (asynchronous I/O) is only allowed to access asynchronous ports. The details of specific logical interfaces relate not to the actual data transfer mechanisms used, but to buffer format. Thus, each interface is constrained to request actual data transfers from the drivers through the kernel.

Because control of the network is centralized, all external routing is under the control of the network Supervisor, which is connected to each node through slot 0. As a result, the Supervisor must both know and approve all internal and external circuit routing.

Multiple Extended Processor (MXP)

MXP extends ISIS-II to a cluster of Engine processors, providing more computing power than is available with a single Engine. A maximum of 15 Engines may be linked into an MXP cluster located at the same site. Up to 1000 feet can separate Engines, which are connected by Ethernet hardware. MXP enables Engines to be joined together using a high-speed Local Area Network (LAN).

MXP uses two special hardware devices. The External Processor Interface (XPI) has its own microprocessor, I/O interface, and private memory area (both RAM and ROM). XPI's satellite Motorola 68000 microprocessor reduces the load on the main Engine CPU. The Port Switch Module (PSM) switches network links from one Engine to another, providing back-up facilities should an Engine fail.

The ISIS/MXP dispatcher regards all Engine processors in the cluster as a single machine. A slot in one Engine under MXP can just as quickly communicate with a neighbor slot in the same Engine as with a slot in a separate MXP Engine. Figure 2-3 shows the extended dispatcher in a small MXP cluster.

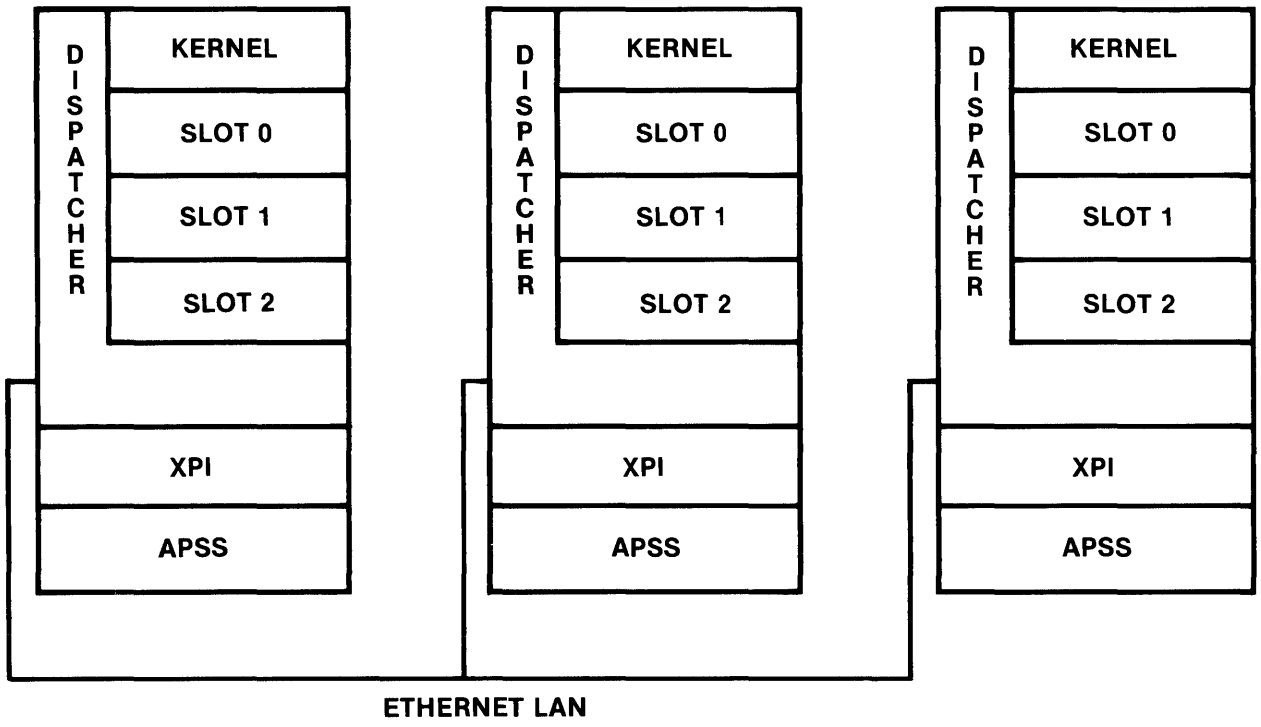


Figure 2-3. A View of an MXP Cluster with Three Engines

Node Code

To be supplied at a later date

Supervisor

CHAPTER 4

SUPERVISOR

CONTENTS

OVERVIEW

The Supervisor is a program that acquires a continuously updated image of global network topology and resource allocation. The Supervisor performs the following:

- centrally controls a TYMNET network
- creates virtual circuits using optimal virtual circuit routing paths
- monitors and allocates resources
- collects critical information on node and host status
- analyzes network diagnostic data and link loading status
- handles global network functions
- frees nodes to handle local processing tasks
- checks usernames and passwords against a Master User Directory (MUD)
- enforces user access controls
- collects accounting information

Network control personnel have access to the Supervisor, PROBE, Tymnet Monitoring and Control System (TMCS), the Network Console, Network Event Monitoring (NEM), and the Network Validations (NETVAL) programs.

The TYMNET Engine on which the Supervisor resides is a normal ISIS-II (Internally Switched Interface System) node. The Supervisor program runs in an ISIS-II job slot, with its six "slave" subprograms in six other slots in the same node. ISIS-II's partitioned slot design allows different processes to coexist in a node alongside the Supervisor. Back-up Supervisors provide a safety margin should the current active Supervisor fail. Periodic rotation of Supervisors keeps all inactive units on permanent "warm" standby. Each Supervisor has its own disk file.

The Supervisor slot contains specific files used in the configuration of a Supervisor node. Many of the slave programs share common parameters with the Supervisor slot. The six slave programs are as follows:

- PROBE is an interactive network monitoring program that serves as an authorized user's interface to the network. Authorized PROBE users can reset Supervisor parameters and shut or reinstate network links and nodes.

- The Error Log slave program has read-only access to the network diagnostic log on disk.

- The Accounting slave program has read-only access to raw accounting data that is stored on the Supervisor's disk.

- The System Message slave provides on-line messages to users about network conditions.

- The Master User Directory slave (UN2) program runs on the Supervisor node and allows the network user validations program (NETVAL) to read and modify the Master User Directory (MUD) file on the Supervisor's disk.

- The Utility slave program on the Supervisor node handles disk-to-disk and tape-to-disk data transfers. The Utility slave can be used in Supervisor configurations and in regular network nodes.

Figure 4-1 shows a Supervisor node in a typical TYMNET network.

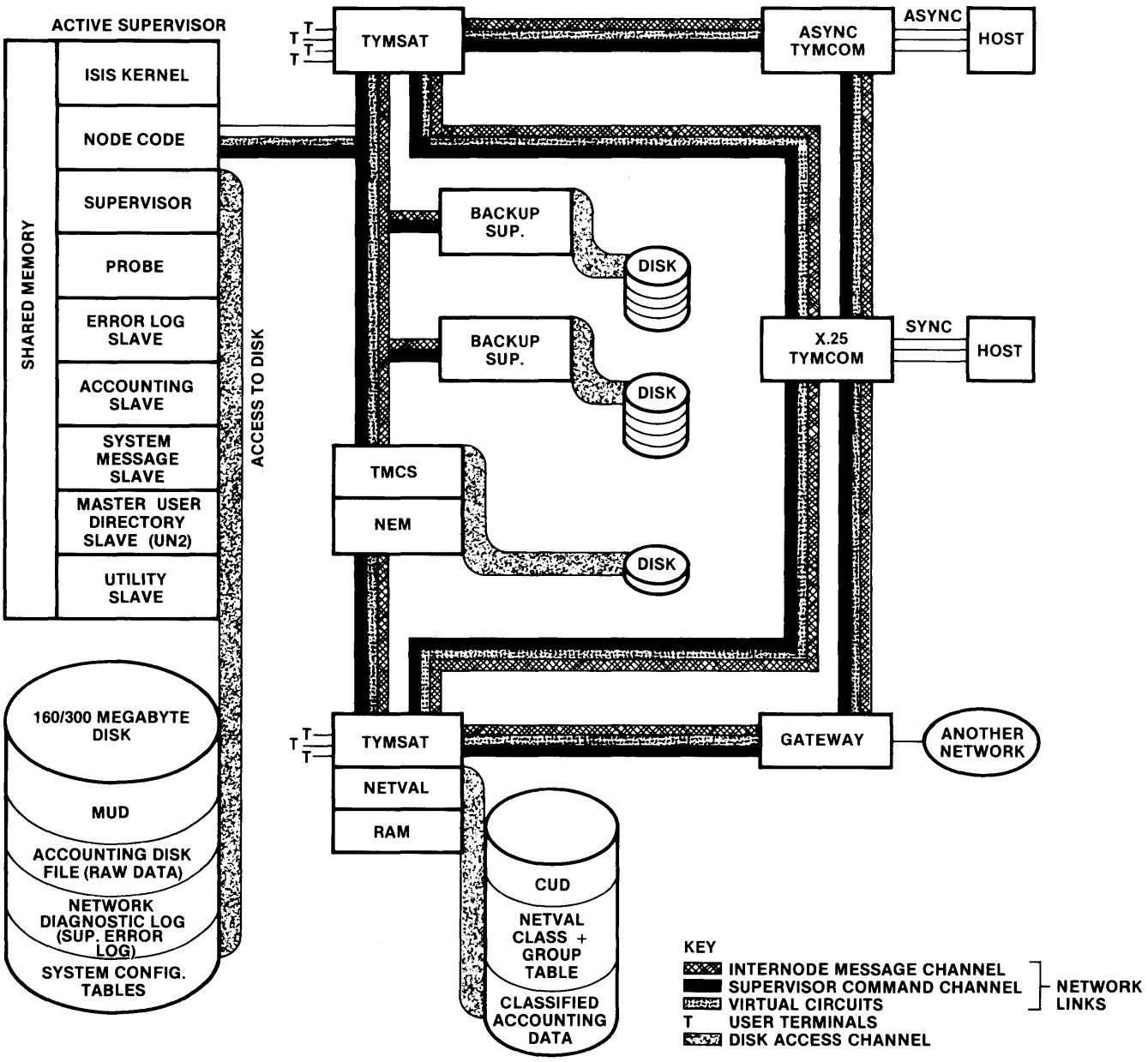


Figure 4-1. A Supervisor Node in a TYMNET Network

NETWORK RESOURCE MONITORING

The Supervisor maintains a current image of network topology and monitors many critical network resources. It is the only location in the network with global network knowledge.

The active Supervisor keeps network topology tables in memory, which consist of node descriptors and host tables. The node descriptors include node numbers, the type and speed of communication links connected to each node, and details of internodal link status. The Supervisor looks at four types of network link information as follows:

- whether a link is up or down (in service or not)
- overloaded
- shut
- if the node itself is out of passthroughs, indicating that no more circuits can be built through it

Host tables contain host numbers, the nodes that are connected to each host, and the current host status.

Personnel at the network control center are able to examine the Supervisor at any time and can receive instantaneous reports on the state of the network. They can also load code into malfunctioning remote Engine Processors and even into the Supervisor itself to rapidly change system parameters.

Using TMCS, special video display monitors show trouble reports that may be color-coded for different urgency levels. A continuous printout may be created on all problem areas, such as overloaded links or malfunctioning and out-of-service nodes. Many system changes can be made from the central network control point.

The Supervisor monitors and allocates resources within the network to ensure even traffic patterns and network balance. Because TYMNET networks have centralized control, these networks overcome the problem of link overloading in a fully distributed type of network. If each node (instead of the Supervisor) makes routing decisions, link overloading and delay can occur because nodes are not aware of routing problems in other parts of the network. Thus, a node could route additional traffic into an already heavily loaded area,

creating a breakdown. However, in a TYMNET network the Supervisor builds new circuits through less heavily loaded areas. In addition, both the Supervisor and network control personnel immediately know the trouble spots.

The Supervisor collects and records network accounting, error, and event information in files on a 160 or 300 megabyte disk. The destination node of a user's virtual circuit sends start and end-of-session messages to the Supervisor. In addition, all nodes send periodic reports to the Supervisor with details of node and link status, as well as reports from ISIS-II slot interfaces.

All data is put into standard format packets that are sent through the network. Nodes check the packets for transmission errors and destination. Using retransmission techniques, TYMNET networks provide error-free data transmission.

If a node is down, severely overloaded, or making excessive retransmissions, the Supervisor becomes aware of the problem through node error reports. In this way the Supervisor monitors each node and link in the network. Error tracking capabilities provide display and central overview functions for network operators, who attempt to remedy the problems.

All Supervisors, active or dormant, incorporate PROBE, so a current picture of network topology is always available to users who log into PROBE on the active Supervisor. Users can receive information on the active Supervisor through the PROBE program. PROBE is an interactive network monitoring program that is part of the Supervisor. PROBE inquiry into an inactive (sleeping) Supervisor determines the past status of the network when that Supervisor was last active (awake).

PROBE itself is a slave program of the Supervisor and can access the diagnostic log (also called the Supervisor error log or the network diagnostic log) and the node descriptor and host tables in the Supervisor's memory. PROBE has a circuit-tracing facility used to instruct the Supervisor to request information from the origin or destination node of a virtual circuit. These nodes generate a list of intermediate nodes in a given virtual circuit and send the list to the Supervisor, which puts it in a buffer accessible to

PROBE. PROBE provides five status levels restricting users to commands authorized by their status limit.

The diagnostic log is a circular disk file containing records of messages from nodes received by the active Supervisor and messages generated by the Supervisor.

ISIS-II slots in nodes send error and diagnostic messages to the node code of each network node, which in turn, sends them to the node code of the active Supervisor along network control channels. The active Supervisor sends the messages to the diagnostic log on disk, which can be read by the Error Log slave program and PROBE. NEM receives messages from the Error Log slave; TMCS receives messages from NEM. Messages in a diagnostic log are grouped chronologically and include the following categories:

- supervisor messages
- network messages
- line status messages
- node reports

PROBE, TMCS and NEM can access (in different ways) the diagnostic data base on disk to retrieve specified information.

Two useful products available for network monitoring and diagnostics are TMCS and NEM. Each product is external to the Supervisor and offers multiple display modes and functions. Both products permit many users with individual user access profiles.

In general, TMCS is similar to PROBE, except it has more features and its own disk files. TMCS performs the following functions:

- permits users to log into PROBE and the Error Log slave program
- maintains a display of network control messages, either color-coded or black and white
- allows the setting of alarms for severe network problems
- provides an overview of the entire network

- maintains a database of physical network data by node and host location and line connection
- builds multiple circuits to NEM and works in conjunction with NEM

NEM is the companion program to TMCS and provides the following functions:

- consolidates error and event messages for all Supervisors, both active and inactive
- provides message retrieval categorized by time, event, node, and host
- builds circuits to the Error Log slave program on each Supervisor node
- asks the Error Log slave to send blocks of data from the diagnostic logs of all network Supervisors
- builds a consolidated database of all network error and event reports
- allows multiple user access to the data and stores up to at least one month's network event history on a circular file

NEM will have a separate disk if located on a different node than TMCS. Both NEM and TMCS can share a common disk if located on the same node.

NETWORK ACCOUNTING RECORDS

TYMNET networks have efficient accounting functions. The Supervisor collects accounting information on network use for user billing purposes through the Raw Accounting Merger (RAM) program. During and at the end of a user's session, accounting data is passed to the Supervisor along special Supervisor control channels by the dispatcher routine in every ISIS-II node. At the beginning of each session, the Supervisor initializes an account by placing an accounting preamble into the accounting file on disk.

RAM is responsible for the following functions:

- collects and processes accounting information from the network Supervisor
- sorts accounting data based on session number, holding the data in memory until all session information is gathered
- writes each completed session record to magnetic tape. (A typical user session record includes login and logout time, input and output character counts, username, and origin and destination node numbers.)
- correlates the data from more than one Supervisor when a new Supervisor takes over the network

USER ACCESS CONTROL

The Supervisor validates usernames and passwords and thus controls user access to the network as well as selects virtual circuit routes at login time. During login, every user specifies the terminal type used, the data transmission speed, and destination host. Terminal location is revealed when a circuit is requested.

NETVAL is used to determine which users can access a network and which origins and destinations can be assessed by validated network users.

NETVAL capabilities are as follows:

- allows a user to update the Controlling User Directory (CUD) on disk
- provides magnetic tape backup for the CUD and restores the CUD from magnetic tape
- updates the MUD of each network Supervisor
- allows users to change their own passwords
- provides privileged and unprivileged user access

The NETVAL contains two databases: the CUD and the NETVAL class and group table. The NETVAL program may reside in an ISIS-II job slot on the same Engine as the network Supervisor for small custom networks. However, in larger custom networks, NETVAL is housed in a separate ISIS-II node.

Each Supervisor has access to a directory of valid usernames and passwords (MUD) which is stored on disk. At login, the user must specify a username, host computer and password. A user login string can be checked instantly against the MUD. An invalid username, password, or user status level or a network access restriction will prevent login. Passwords are ciphered (encrypted) before being placed in the MUD. If someone illegally gains access to a user's MUD entry, the password cannot be determined.

Changes are not made directly to the MUD file by users. NETVAL updates a database (CUD) which is almost identical to the MUD. The CUD resides on the same node as NETVAL. The purpose of NETVAL is to make sure the MUD is the same as the CUD. NETVAL uses two processes to accomplish this: the immediate MUD update and the daily consistency check. MUD updates are performed on the MUD as

soon as changes are made to the CUD. In a consistency check, a circuit is built from NETVAL to the slot housing the Master User Directory slave (UN2) on the Supervisor. UN2 has read/write MUD access and compares each block of data on the MUD to a similar block on the CUD. If differences are found, NETVAL copies the CUD block to the MUD.

Users often work with a particular "home host", which can be specified or removed through the Supervisor by updating the user profile. Alternative hosts can be permitted for individuals or groups.

Identification information is presented as follows:

- username which identifies a user or group of users
- optional host number (destination address)
- optional subhost address (an extension of the destination address)
- security password

The Supervisor can implement Closed User Groups (CUGs) to control access to the network by defining a user's network entry and destination points in the user's access profile in the MUD. This profile contains definitions which specify entry points from which the user may originate circuits (origins) and hosts to which he may build circuits (destinations).

A NETVAL file called the Class and Group Table, referenced by the Supervisor when checking a user's access status, contains a complete list of hosts and nodes.

NETWORK SECURITY

The Supervisor's centralized control offers a high level of network security. Because network control is extremely effective, it is improbable that a TYMNET network could be entered by an unauthorized user and highly unlikely for the Supervisor to be illegally accessed. Unlike TYMNET, fully distributed networks are not centrally controlled; therefore, each node has an operational map of the network and a list of valid users. Illegal access to one node can open the entire network.

In centrally controlled networks such as TYMNET, each node is only a small part of the network and has limited autonomy, no network-wide map, and a limited role in security. Embedded nonalphanumeric characters in the user password and a complex password ciphering algorithm in the MUD provide extra security safeguards. TYMNET is the only network with such a sophisticated password security feature. In addition, after a user makes three unsuccessful attempts to login, the user is prevented from further network activity. This safeguard prevents automated login attempts using a microcomputer and a variety of passwords. A Supervisor error log entry is created for all login attempts greater than three.

OPTIMAL NETWORK CIRCUIT PATHS

The Supervisor creates a source-to-destination virtual circuit path through the network based on the most economical use of network resources.

The Supervisor maintains network topology tables comprising node descriptors and host tables that contain node information, such as which hosts are connected to each node, available network lines and their speeds, and overloaded links.

All links in the circuit are assigned resource costs that increase for slow or overloaded links and become infinite for shut, operator closed, out, or failed links. See Table 4-1 for an example of resource costs.

Table 4-1. Resource Cost Table

Link Speed KBPS	Normal Link	Link Overloaded in One Direction	Link Overloaded in Both Directions
4.8	36	73	109
9.6	30	61	91
56	26	53	79

In unison with the above table, Figure 4-2 shows a circuit cost example. The Supervisor calculates the most economical path from terminal to terminal, terminal to host, host to terminal (or printer), or host to host based on the use of network resources. Since data communications, even to satellites, are fast, it does not matter if the selected route is not the shortest physical distance from source to destination. The calculation does not involve distance or dollar costs, only network resources used. A longer but faster link may be more efficient if network resources are used for a shorter time. The Supervisor has the necessary information to make routing decisions since it alone has the complete network picture.

The least-cost path from the terminal to the host uses nodes A, B, and D. The resource cost is AB + BD = 30 + 30 = 60. The cost is higher on the direct route AD, which is overloaded and costs 53 network resource units.

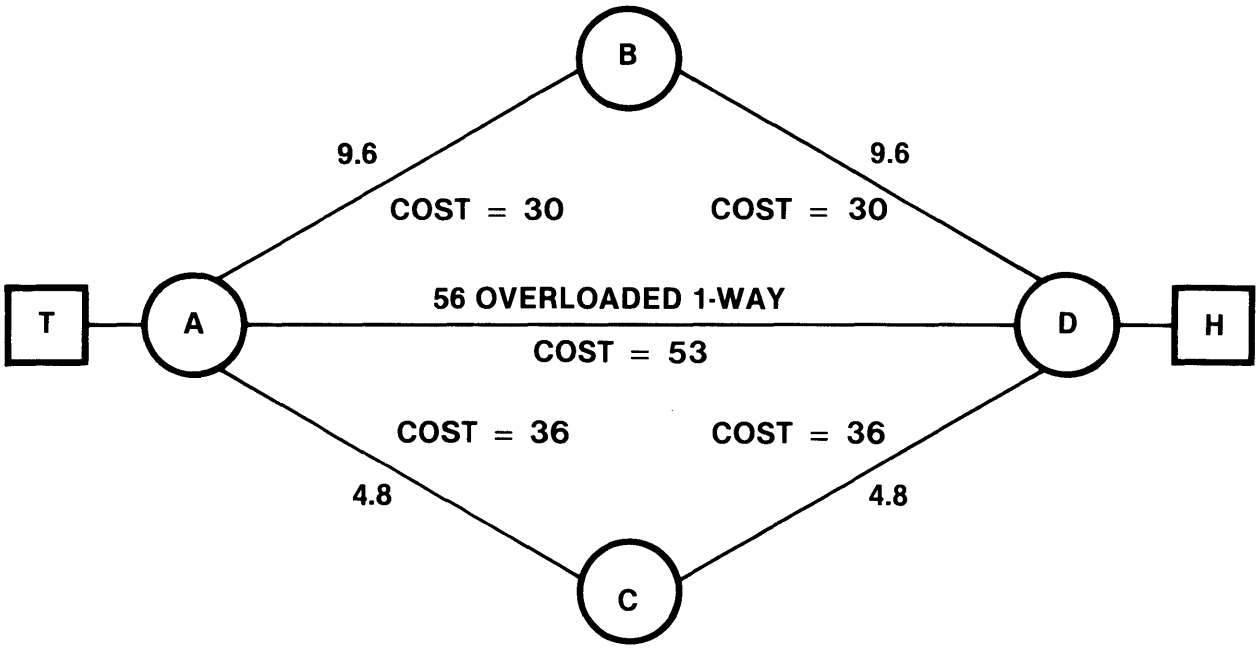


Figure 4-2. Sample Circuit Cost

CIRCUIT ROUTING INFORMATION TO NODES

After a user has logged in and the Supervisor has validated the login and username, the Supervisor calculates the least resource cost path along network links from source to destination. It then sends the origin node a special message, called a needle, containing the path for the circuit. Once the Supervisor sends the needle to the first node, the Supervisor is no longer involved in circuit building.

The needle contains a list (created by the Supervisor) of nodes on the virtual circuit path, and it travels along regular synchronous network links to each node on its list. Each node builds its part of the circuit using its permuter tables and buffer pairs. The needle threads its way through the network creating a virtual circuit behind it. At the destination node, having completed its task, the needle disappears. The destination node sends a needle acknowledgement message (not the original needle) to the Supervisor indicating that the entire circuit is complete. Once established, the circuit is used for data transmission. The Supervisor then merely monitors the circuit through accounting and status reports from nodes.

The needle contains the following information:

- a numbered invoice, consisting of the session number and the active Supervisor number, used for accounting purposes
- origin and destination node and port numbers
- a list of the nodes comprising the circuit's path
- the destination host number
- a username and terminal identifier
- flags indicating the circuit type and speed

Figure 4-3 shows the circuit building process. A user at the terminal (lower left) logs in to the host (lower right). The virtual circuit is built through nodes A, B, and C in the center.

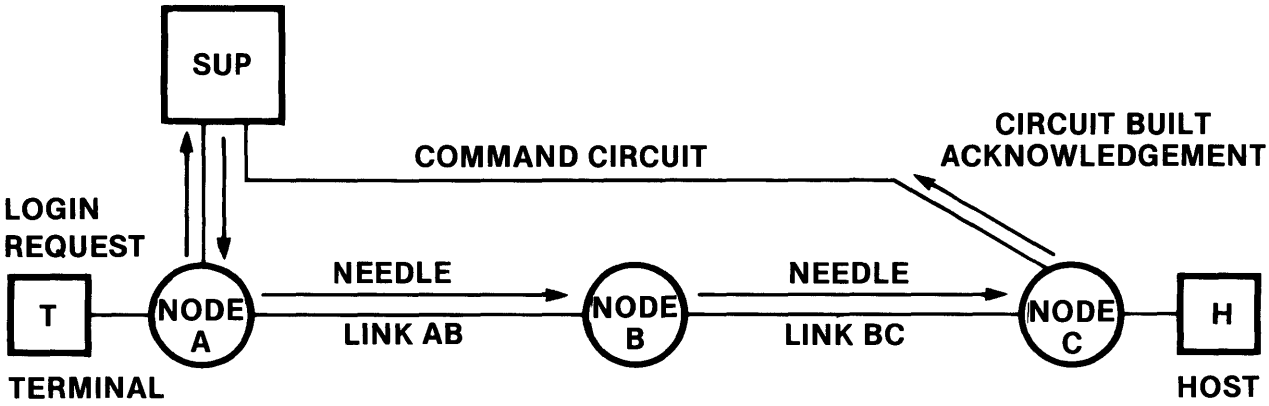


Figure 4-3. Network Circuit Building

The originating node, A, receives the needle from the Supervisor and checks the node number of the next node in the path. Node A assigns an unused channel on the regular network link and sends the needle to the next node, B, after crossing off its own node number, A, from the needle's list.

The next node, B, receives the needle on an unassigned channel, and checks if node B is the destination node. If not, node B passes the needle on to node C, the next node on the needle's list.

When the destination node C is reached, node C checks the status of the attached destination host. If the host is available, the node completes the circuit; if the host is unavailable, the circuit is cleared and a system message sent by the Supervisors System Message slave appears on the user's screen.

SUPERVISOR TAKEOVER

Depending on the arrangements made, at 24 hour intervals, the active Supervisor in a TYMNET network is alternated with a standby unit by network control personnel. This routine helps keep all Supervisors in full operating condition and typically occurs at times of low network traffic, such as 4:00 a.m. The same procedure is used to activate a standby Supervisor if an active Supervisor fails. Supervisor takeover does not affect existing circuits. However, new logins (and circuits) will be temporarily unavailable during a short period, until the new Supervisor takes over.

Standby Supervisors are kept in an inactive or "sleeping" state by a series of commands, called "sleeping pills," that are given every 20 seconds by the active Supervisor. Standby Supervisors are held at graded depths of sleep, with the least sleepy Supervisor waking first. If a sleeping Supervisor fails to get a scheduled sleeping pill, it will wake and begin to take control of the network. A little later, another sleeping Supervisor will wake if it too has not received its sleeping pill and will also attempt to take over the network. The newly active Supervisor must immediately send sleeping pills to inactivate all other Supervisors.

When a Supervisor wakes, it knows nothing of current network conditions. The sleeping Supervisors do not monitor the network. Data in a dormant Supervisor's diagnostic error log is not erased when the Supervisor becomes active, but will eventually be overwritten by subsequent data. When a Supervisor awakes, it takes over the nodes nearest itself by sending takeover commands to the nodes over the Supervisor's control channel. The Supervisor then takes over the next nearest nodes in turn, until it controls the entire network. For even the largest custom networks, takeover time is less than a minute. Each node that is taken over sends to the active Supervisor a list of neighbor nodes, active links, their speeds and available passthrough buffers, and host numbers. In this way, an image of the network is created in the memory of the new active Supervisor.

Supervisors are given numbers which indicate takeover priority for an active Supervisor. Supervisors with lower numbers have highest priority. If two Supervisors were to waken simultaneously and each were to take over the network, nodes would respond to takeover messages from the Supervisor with the lower number. The lower priority Supervisor is returned to a sleep state, resolving any possibility of a Supervisor fight. A Supervisor's drowsiness factor can be changed by network personnel.

TYMSAT

CHAPTER 5

TYMSAT

CONTENTS

OVERVIEW

The TYMSAT is the asynchronous terminal interface to the TYMNET network. This interface translates asynchronous terminal communications into the synchronous, packet-switching protocol that the network handles.

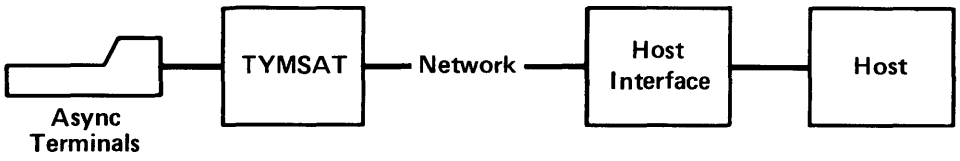


Figure 5-1. The TYMSAT Interface

The TYMSAT has two distinct implementations: the stand-alone TYMSAT node and the TYMSAT that runs in a job slot on an Internally Switched Interface System (ISIS) node. The differences between these two implementations will be pointed out where applicable in this document.

The TYMSAT communicates with terminals through ports, which are the physical and logical channels through which data passes. A port or pseudo-port exists for each terminal connection. Ports are logically defined in the configuration file (Tymfile). They can be defined to provide information to the user, such as login and error messages, and to send login information to the host for the user.

Several kinds of ports can handle different kinds of asynchronous communication. On the ISIS TYMSAT, Serial Input/Output (SIO) ports are based on a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) chip. Printer ports handle output to high-speed printers.

On both the ISIS and stand-alone TYMSATs, Permanent Virtual Circuit (PVC) ports are designed to handle a number of complex processes, including login. Parameters in the Tymfile configure PVC ports to perform these processes. For example, two such parameters, Automatic Terminal Identification (AID) and the Automatic Login Option (ALO), enable PVCs to identify terminal type and user to the system automatically when the terminal is first turned on. Other PVC parameters enable data

from specialized terminals to pass through the TYMSAT to the network. Thus, PVC ports can handle Telex, LISA (voice response), and Videotext terminals.

A special subset of PVC ports, Multiplexed Permanent Virtual Circuit (MPVC) ports, allow economical network use by enabling several terminal users to share the same network circuit. MPVCs use pseudo-ports, with data from several real ports actually occupying one multiplexed circuit.

THE CONSAT PROGRAM

The software for the TYMSAT version described in this document is called the CONSAT (Consolidated TYMSAT). Wherever the name TYMSAT appears, it refers to the CONSAT version, unless specifically noted otherwise. This TYMSAT differs from past versions in that the same source code can run either in a stand-alone TYMNET node or on an ISIS node.

The stand-alone implementation is a TYMNET node dedicated to the TYMSAT function. This node runs the CONSAT and Node Code functions as an integrated system.

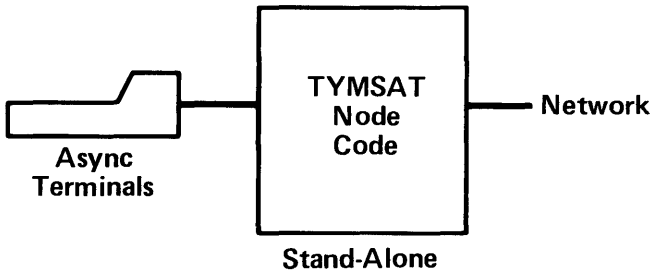


Figure 5-2. Stand-Alone TYMSAT

The ISIS implementation of the TYMSAT runs in a slot on a TYMNET node. Node Code must be loaded in slot zero. Other TYMNET programs can also be loaded on the same node.

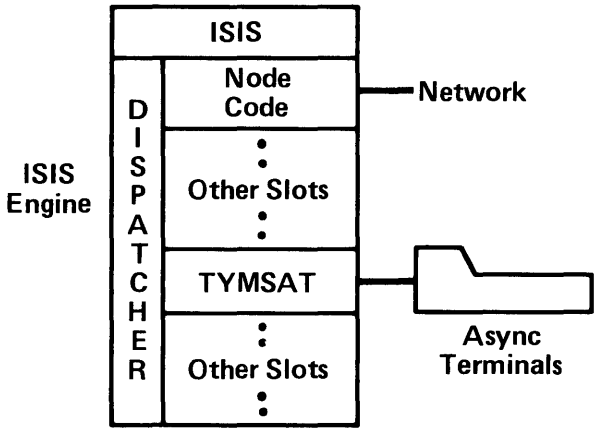


Figure 5-3. ISIS TYMSAT

The stand-alone TYMSAT is generally preferable in an environment consisting of single application, heavy-volume traffic. If the environment demands several applications on a single TYMNET node, the TYMSAT running under ISIS is required.

SUPPORT CAPABILITIES AND CONSTRAINTS

The TYMSAT communicates with terminals over asynchronous lines and physical ports using asynchronous input/output cards, Serial Input/Output (SIO) motherboard cards, and printer cards. If present, printer cards are automatically assigned to logical port 0 and immediate successors by the TYMSAT.

Support capabilities differ between the ISIS and stand-alone implementations of the TYMSAT. However, they have major features in common. Both implementations support most asynchronous terminals and printers, including Telex, and they support ASCII terminals in the 110-1200 baud or 300-4800 baud range.

STAND-ALONE TYMSAT SUPPORT

The stand-alone TYMSAT supports Permanent Virtual Circuits (PVCs), Multiplexed PVCs (MPVCs), and Telex "Extension Cord" capabilities. Although it has fewer capabilities than the ISIS version, the stand-alone (or "solo") TYMSAT compensates with increased performance.

The following table summarizes stand-alone TYMSAT support.

Table 5-1. Solo TYMSAT Support

Supported	Unsupported
asynchronous	addressable ports
MPVCs	Baudot (except 50-baud, for Telex only)
printer ports	LISA terminals
PVCs	SIO ports
Telex terminals	Videotext terminals

ISIS TYMSAT SUPPORT

The ISIS TYMSAT has all the capabilities of the solo version. In addition, it supports SIO ports, addressable ports, LISA terminals, the X.28 Packet Assembler/Disassembler (PAD), Videotext terminals, and Baudot terminals.

The ISIS TYMSAT supports the following baud rates for SIO ports:

50	1200
75	2400
110	4800
300	9600

The ISIS TYMSAT supports Baudot terminals at both 50 and 75 baud. The 50-baud rate is supported for both asynchronous and SIO ports. The 75-baud rate is only supported for SIO ports. Telex terminals use 50-baud asynchronous Baudot.

The following table summarizes ISIS TYMSAT support.

Table 5-2. ISIS TYMSAT Support

Capabilities	Constraints
addressable ports	MPVCs cannot be configured for port addressing
asynchronous terminals	
Baudot terminals (50 & 75 baud)	75-baud Baudot terminals must use SIO ports
LISA terminals	
MPVCs	
printer ports	
PVCs	
SIO ports	
Telex terminals	
Videotext terminals	
X.28 PAD	

SYSTEM OPERATION

The TYMSAT is the interface that connects the user's asynchronous terminal and the network. To make this connection, the TYMSAT uses login information, which can be either provided by the user as part of a login string or configured into the system at generation time. Login information consists of the following:

1. Terminal Identifier (TID). The single-character TID identifies specific terminal operating chacteristics to the TYMSAT. The user enters the TID upon login. The operating characteristics defined by the TID are baud rate, character set, carriage return delay, and echoing requirements.

2. Control Characters. The user can specify operating characteristics for a session by imbedding control characters within the login string. These characters can be used to control throughput rate, output parity, and other variables that the user may need to change from session to session.

3. Username. The user enters username in response to the system prompt, "PLEASE LOG IN:".

4. Host Destination. This is specified as a number. The user must be valid on the destination host.

5. Password. The user enters a password to access password-protected usernames.

SYSTEM CONFIGURATION

When the TYMSAT is generated, a number of files are assembled together in order; the particular set of files used is environment-dependent.

For the stand-alone TYMSAT, eight files are involved in system generation. These files are listed below with the syntax for their names.

```
xxxx   =  node number
vv.rr  =  version number and revision level
```

1. command file (NDxxxx.CMD)
2. CONSAT initialization file (ISCSvv.Irr)
3. Node Code initialization file (TIIvv.Irr)
4. Tymfile (NDxxxx.TYM)
5. Node Code source file, part one (TIIvv.Rrr)
6. CONSAT source file (ISCSvv.Rrr)
7. Node Code source file, part two (TIIvv.Frr)
8. Bound file (NDxxxx.BND)

The command file contains pointers to the next six files: the two initialization files, the configuration file (called the Tymfile), and the three source files. The command file is used by the Node Assembler/Debugger (NAD) program to assemble these files into the total load module for the Engine. This load module is a machine-language file called the Bound (BND) file.

For the ISIS TYMSAT, five files are involved in system generation. These files are listed below with the syntax for their names.

```
xxxx   =  node number
ss     =  slot number
vv.rr  =  version number and revision level
```

1. command file (NDxxxx.Css)
2. initialization file (ISCSvv.Irr)
3. Tymfile (NDxxxx.Tss)
4. CONSAT source file (ISCSvv.Rrr)
5. NIB file (NDxxxx.Nss)

The command file contains pointers to the next three files: the initialization file, the Tymfile, and the CONSAT source file. The command file is used by the NAD program to assemble the machine-language version of the code contained in the other files. This machine-language file is called the NAD Image Binary (NIB) file.

Of the system files discussed above, the Tymfile is important because it is configured by the system programmer or field engineer to conform to the specific installation. The Tymfile contains the macros that specify TYMSAT configuration and constraints. These features and options are set at system generation. Interactive access to the TYMSAT for changing options is not possible while the program is running.

SPECIAL FEATURES

The TYMSAT supports special communication capabilities, such as the X.28 Packet Assembler/Disassembler (PAD) and port addressing, and supports special terminals, such as Telex, LISA (voice response), and Videotext.

On the ISIS TYMSAT, the PAD capability allows asynchronous Data Terminal Equipment (DTE) to communicate through the TYMNET network to an X.25 host.

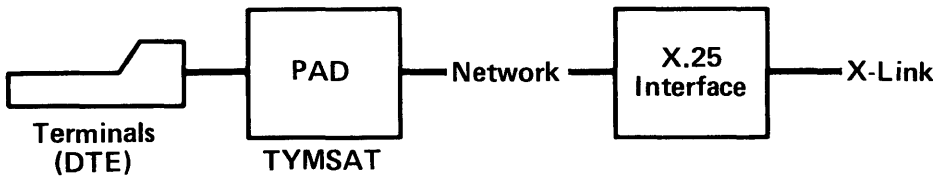


Figure 5-4. TYMSAT with PAD

The X.25 Interface can occupy a slot on the same ISIS node as the TYMSAT, or it can reside on a different TYMNET node.

Another TYMSAT enhancement, port addressing, enables a terminal user to send data to an asynchronous port that has an assigned host number. The receiving port acts as a host in this interchange by accepting data from another terminal, but it retains its capacity to send data as well. The TYMSAT allows port addressing between asynchronous personal computers, but does not provide echoing to the display screen.

The TYMSAT Telex-terminal support is called "Extension Cord" because it enables a Telex terminal to communicate through the TYMSAT to the Telex Gateway. The Gateway in turn sends the data to a Telex switch, which accesses the Telex network.

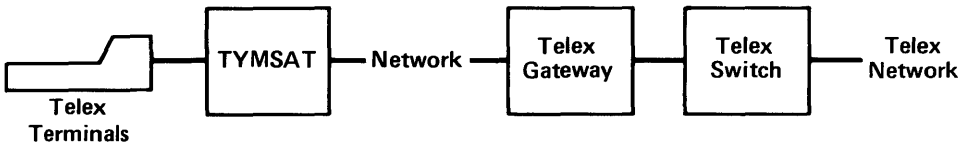


Figure 5-5. TYMSAT/Telex Extension Cord

The Telex Extension Cord is supported on both the stand-alone and ISIS TYMSATs. The Telex Gateway can occupy a slot on the same ISIS node as the TYMSAT or it can reside on a different TYMNET node.

Both LISA and Videotext terminals can connect to the network through an ISIS TYMSAT. The TYMSAT Tymfile contains a Permanent Virtual Circuit (PVC) parameter, called Automatic Identification (AID), that allows these terminals to send and receive data transparently through the network.

X.25

To be supplied at a later date

ISIS TYMCOM

CONTENTS

OVERVIEW

A TYMCOM is a TYMNET network interface that may be connected to a user's host computer. An asynchronous TYMNET Internally Switched Interface System (ISIS) TYMCOM Interface provides a set of individual asynchronous line interfaces to an attached host computer. This interface appears to a host computer as a bank of asynchronous dial-up modems or asynchronous terminals, depending on the configuration. A TYMCOM is usually located on-site with host computer hardware. Each TYMCOM interface runs in an ISIS slot in a TYMNET node.

The ISIS TYMCOM communicates with a host computer that uses asynchronous I/O cards (asynchronous ports) and SIO mother board cards (SIO ports). Asynchronous ports use Electronics Industry Association (EIA) RS-232C signal protocol supporting 8-bit or 5-bit characters. SIO ports use EIA RS-232C signal protocol supporting 8-bit characters or use a subset of X.20 protocol supporting 5-bit characters.

The ISIS TYMCOM allows a maximum of 128 network hosts, although the Supervisor can restrict this to a smaller number. A maximum of 128 subhosts can be specified for the entire TYMCOM. The number of subhosts can be greater than the ports, as ports can be reassigned dynamically using the Tymcom Operations Manager (TOM).

The ISIS TYMCOM allows a maximum of 128 asynchronous and Serial Input/Output (SIO) ports. Asynchronous ports and SIO ports can be configured in the same job slot, allowing greater flexibility.

The ISIS TYMCOM provides a stored message facility that allows a message to be displayed on terminals when a host is down or when a port does not answer.

Figure 7-1 illustrates a TYMNET Asynchronous ISIS TYMCOM Interface.

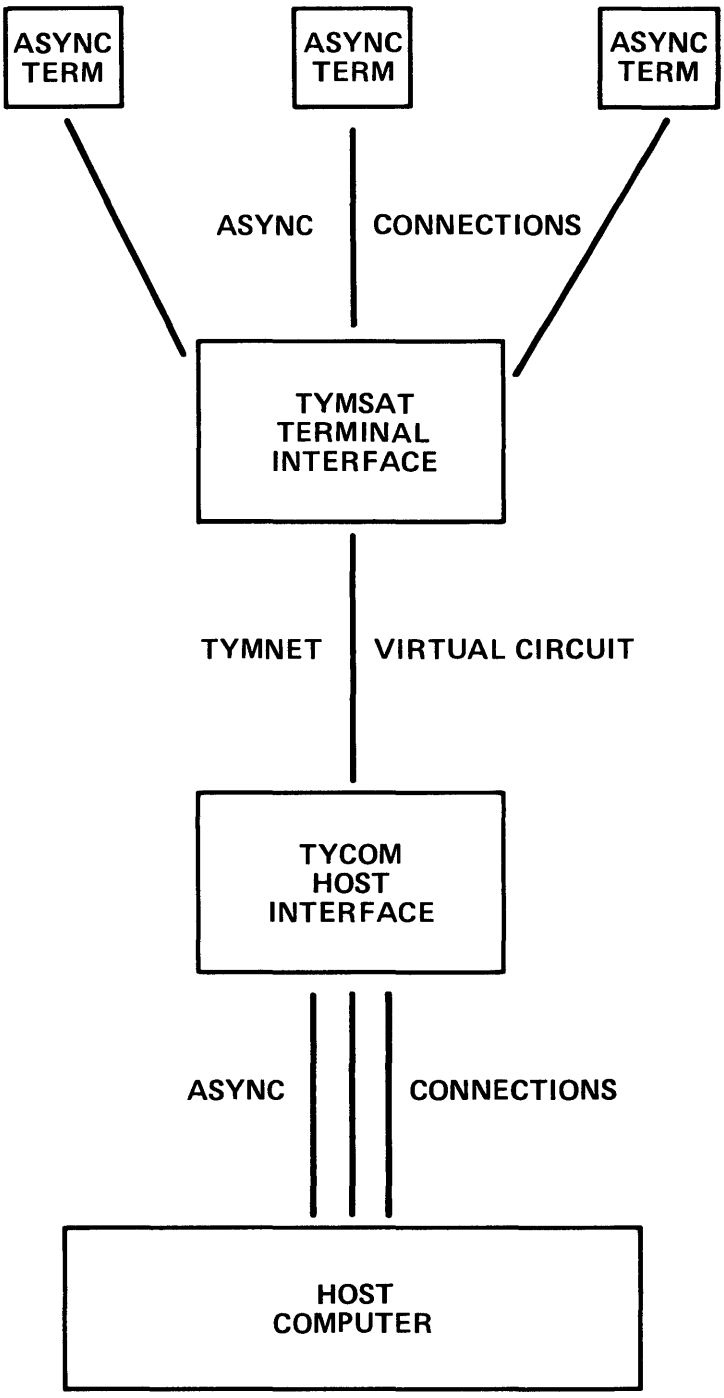


Figure 7-1. Asynchronous ISIS TYMCOM Interface

Signal Protocol

The ISIS TYMCOM supports the following host computer communication line speeds.

Table 7-1. Communication Line Speeds

Range	Line Type
75-4800 baud	asynchronous
75-9600 baud	SIO

Various cable pin configurations provide capabilities of SIO lines employing EIA RS-232C signal protocol subsets and X.20 subset signaling standards. Options provide other capabilities when generating TYMCOM software. On heavily loaded machines, the overall throughput capability of the slot may occasionally limit the character rate of individual ports to less than full capacity. Also, the origination (terminal) and network access may limit the overall throughput capability.

Software Signal Protocol

The ISIS TYMCOM has the ability to detect two control signals from the host. It can also manipulate two control signals toward the host for each port. The detected control signals are selected by the cable pin configuration on a port-by-port basis and are determined by the mode in which the TYMCOM operates. The TYMCOM is used as Data Communications Equipment (DCE) and appears to the host as a bank of modems. The TYMCOM can also be used as Data Terminal Equipment (DTE) and appears to the host as several hardwired terminals.

Hardware Signal Protocol

The signals that are actually presented to and received from the host can be varied by the selection of cable options. There are two standard pin configurations available. Special cables or cable adaptors may be ordered to suit customer requirements as long as software protocol is satisfied.

The first cable pin configuration is used for host ports that expect to communicate with a terminal using a modem. The TYMCOM appears to the host as a modem.

The second cable pin configuration option is used for host ports that communicate with a terminal over a hardwired connection. The software operates exactly the same; the only difference is in the cable.

SYSTEM GENERATION

A concise system generation command language provides the user with complete control of the TYMCOM environment. Most configuration options have defaults and are not specified unless a change occurs. This configuration feature enables a user to create a fast and efficient TYMCOM or one suited for a field test environment in which all specified options allow high visibility of minor changes.

The TYMCOM has the following four option levels:

global	options that affect the entire TYMCOM
host	options that affect all subhosts associated with a network host number
subhost	options that affect only the given subhost
port	options that affect each individual port

TYMCOM OPERATIONS MANAGER

Many ISIS TYMCOM options can be displayed and manipulated using the TYMCOM Operations Manager (TOM). TOM is a TYMCOM interactive monitoring and control process. It is accessed through a hardwired port on a TYMNET Engine or through a network virtual circuit. The number of hardwired ports and positions can be changed. However, the maximum number of users of TOM at any one time is specified during system generation.

A terminal connected to the TOM receives all diagnostic output messages from the TYMCOM. This terminal can be used to receive status reports concerning host processes and slot ports.

2780/3780/HASP Interface

CHAPTER 8

2780/3780/HASP INTERFACE

CONTENTS

OVERVIEW

The TYMNET 2780/3780/HASP Interface provides access to customer hosts through the TYMNET network. It provides support for the following Bisynchronous (BSC) protocols:

- 2780 native mode interface
- 3780 native mode interface
- HASP multileaving protocol
- Virtual terminal mode interface
- DOS/MLI interface
- RSCS/VMB interface
- Transparent bisynchronous interface

The Houston Automatic Spooling Program (HASP) provides supplementary job management, data management, and task management functions such as job flow control, task organization, and spool capabilities. The term HASP is equated with a multileaving protocol.

Tymnet's special purpose operating system for communication processing, the Internally Switched Interface System (ISIS), provides a multiprogramming environment for the operation of several communications interface programs. Each 2780/3780/HASP interface runs in an ISIS slot in the same or in another TYMNET Engine.

Hardware and software changes need not be made at the host computer or at the remote terminal. Switched (a telephone network with dial-up access, or a direct connection) or nonswitched (a leased or direct connection) host ports and terminal ports are supported.

Flexible routing through the TYMNET network requires an extra operational step when specifying the network destination. This is available for 2780/3780/HASP native mode, virtual terminal mode, and is called the Switched Virtual Circuit (SVC) facility. The terminal user can submit the network signon record from the user's operating console, card reader, or other input devices which are allowed to emulate the card reader function by the terminal system.

Optionally, routing through the TYMNET network to a dedicated host or terminal can be accomplished without an extra operation step. In this case, the interface establishes the network connection without any user interaction. This Permanent Virtual Circuit (PVC) facility is available for all modes.

Table 8-1 illustrates the network services for each interface facility.

Table 8-1. Network Services

Interface Facility	Native Mode	Virtual Mode	PVC	SVC
2780	X	X	X	X
3780	X	X	X	X
HASP	X		X	X
DOS/MLI	X		X	
RSCS/VMB	X		X	
Transparent BSC	X		X	

Network services common to all interface facilities are the following:

- switched/leased line connection
- hardwired connection
- half duplex transmission
- full duplex transmission
- permanent virtual circuit

Reliability, Availability, and Serviceability

System availability is enhanced by automatic PVC redirection provided by the TYMNET network Supervisor based on PVC username validation.

Serviceability is high due to debugging and monitoring facilities installed within the system. The 2780/3780/HASP interface incorporates a powerful dynamic debugging process which is accessible through the node's kernel host. This tool provides the ability to monitor in real-time the following items:

- received and transmit operation codes
- ISIS dispatcher messages
- overall line state

2780/3780 Link Level Protocol

Data links between the host and host port, and the terminal and terminal port, use BSC point-to-point protocol.

Both transparent and nontransparent text modes are supported. The following text modes are not supported:

- ASCII or 6-bit data link control
- leading graphics
- limited conversation mode

The interface can transmit data frames that require more than one second of transmission time at the rated line speed. Although IBM specifications require that synchronous characters be inserted when transmission time exceeds one second, the interface does not insert synchronous characters.

HASP Link Level Protocol

Data links between the host and host port, and the terminal and terminal port use BSC multileaving protocol. Both transparent and nontransparent text modes are supported.

The interfaces can transmit data frames that require more than one second of transmission time at the rated line speed. Although IBM specifications require the insertion of synchronous characters when transmission exceeds one second, the interfaces do not insert synchronous characters.

SUPPORT APPLICATION

TYMNET 2780/3780 Terminal and Host Interfaces operate in native mode. Figure 8-1 illustrates a native mode operation.

The diagram illustrates two TYMNET 2780/3780/HASP Interfaces connected by virtual circuits using a TYMNET network. The top part of the diagram shows two host ports and one terminal port within the same interface. The ports supply transmit and receive clocks; crossover cables are used. Terminals are connected to the host by TYMNET virtual circuits between the 2780/3780/HASP interfaces.

The bottom part of the diagram shows that both switched (dial-up) and nonswitched (leased) lines can be supported by the same interface. Only one terminal can access a switched terminal port at a time. A 208B is a modem connected to a dial-up line. On a leased connection, the 208A modem supplies the clocks; normal cable is used.

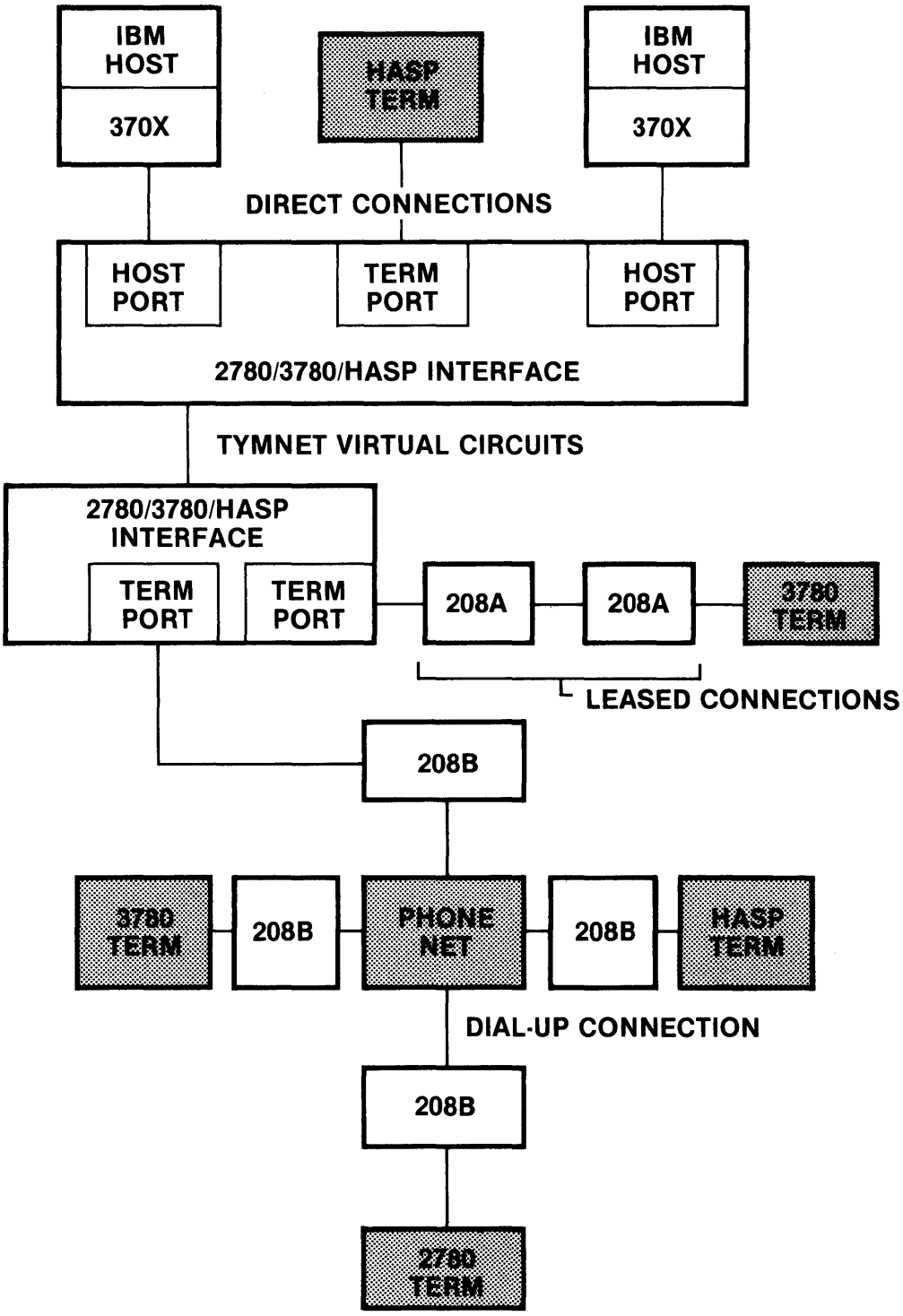


Figure 8-1. 2780/3780 Interface Native Mode

External Support Capabilities

External support capabilities for the 2780/3780/HASP interface include hosts and terminals. The interface is compatible with IBM hardware and software, and supports the following remote terminals:

- 2770
- 2780
- 3770
- 3780
- HASP

The 2780/3780 interface can also support non-IBM hosts with equivalent capabilities. Contact a Tymnet representative to verify compatibility with a specific non-IBM terminal if variations in non-IBM vendor's implementations occur.

The interface type must be specified for each line. Switched and nonswitched terminal interface ports are supported.

2780/3780 VIRTUAL TERMINAL MODE

The TYMNET 2780/3780 Interface virtual terminal mode allows EBCDIC 2780/3780 BSC protocol devices to communicate with ASCII asynchronous character mode hosts.

Features of virtual terminal mode are as follows:

- The 2780/3780/HASP native mode interface, to which virtual terminal mode is added, supports standard EBCDIC 2780/3780 protocol and physical layer signaling including DTR/DSR, RTS/CTS, and RING.

- The interface has a flexible signon procedure which allows a connection to ASCII asynchronous character mode hosts. A virtual signon command allows activation of virtual terminal mode on ports with that feature enabled. An alternative symbolic login command allows selection of either native mode or virtual mode hosts based on a predefined login string located in the interface and referenced by the symbolic command. The predefined string specifies whether the target host is native or virtual mode and may include additional data to be sent to the host upon call completion. This feature can be used to eliminate a second login string to the target host.

- The interface has a flexible code conversion scheme which processes any special requirements on a character-by-character and call-by-call basis.

- An efficient blocking and deblocking scheme for outgoing and incoming data is provided. This is processed by specifying buffer sizes, special break characters, and timeout values during system generation.

DOS/MLI INTERFACE

The TYMNET DOS/MLI Interface is based on the TYMNET 2780/3780/HASP Interface. Availability is enhanced by automatic PVC redirection. This facility is provided for all PVC products by the TYMNET network Supervisor based on PVC username validation.

Figure 8-2 illustrates two remote DOS/MLI systems. Each system is connected to a switch located in front of a host DOS/MLI system. DOS/MLI protocol is used over the communications links to allow spooling of data between the two systems.

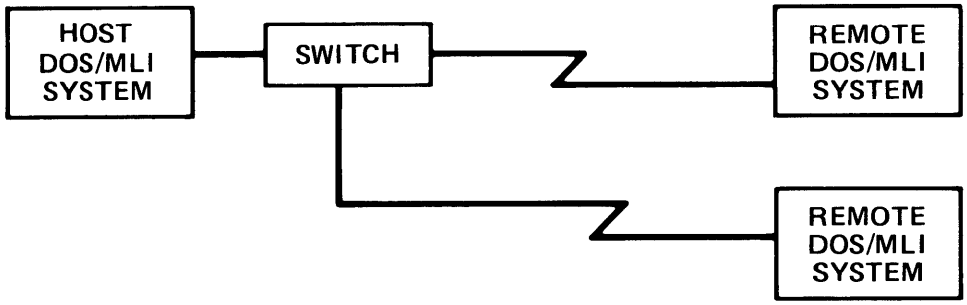


Figure 8-2. Application of Standard DOS/MLI Support

Figure 8-3 illustrates the application of the TYMNET DOS/MLI Interface to connect two remote DOS/MLI systems to a single DOS/MLI host system. Although the figure shows both remote systems simultaneously connected, only one remote system can be connected to the host at any one time.

Multiple virtual circuits are used between the host and the remote interface. Console and input streams share one master virtual circuit. Printer and punch streams use separate virtual circuits. The separation of traffic onto separate circuits allows for easily managed flow control for each system.

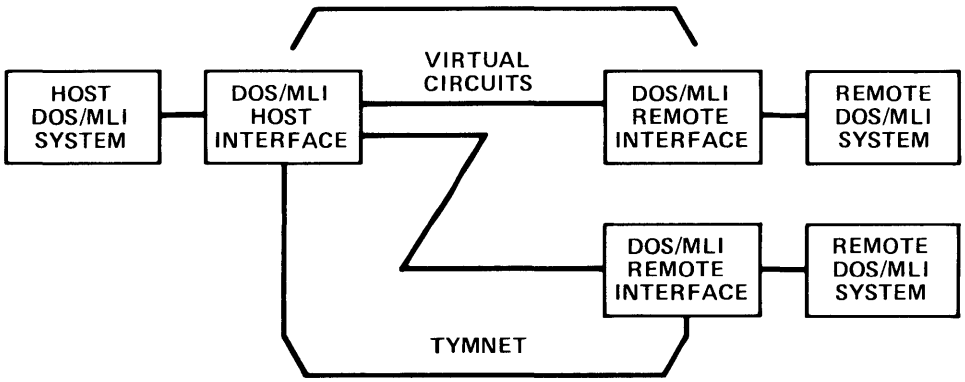


Figure 8-3. TYMNET DOS/MLI Support Application

Features of the DOS/MLI interface are as follows:

- The interface supports a maximum of eight 9600 bps links using standard synchronous ports.
- The interface provides throughput comparable to a direct connection. Local acknowledgments are used to insure continuous flow through the network. Recoverable errors are handled locally.
- The DOS/MLI interface conforms to protocol procedures as defined in the IBM VSE/Power Program and Logic Manual, 5746-XE3, and by actual DOS/MLI session analysis.
- A single DOS/MLI host system supports multiple remote DOS/MLI systems. The remote interface captures the remote ID from the signon record and passes it to the host DOS/MLI system using the DOS/MLI host interface.
- Remote and host lines are supported by the same interface.
- 2780/3780/HASP protocols are supported on other lines on the same interface.
- Maximum block size, maximum number of records per block, and maximum number of output streams are interface generation parameters.

RSCS/VMB INTERFACE

The TYMNET RSCS/VMB Interface is designed to replace existing leased-line VMB networks with a Tymnet-based network product. Automatic PVC redirection is provided by the TYMNET network Supervisor based on username validation. Every TYMNET VMB Interface has the capability to accept or originate PVCs.

Figure 8-4 illustrates two VM/RSCS systems with associated front-end processors connected using a communications link and two data sets. VMB protocol is used over the link to allow users logged into the VM/RSCS systems to spool data between the two systems.

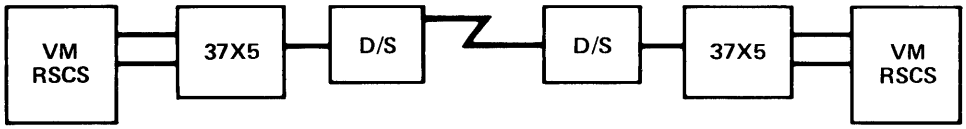


Figure 8-4. Standard VMB Support Application

Figure 8-5 illustrates the application of the TYMNET VMB Interface to connect two VM/RSCS systems. Each link attached to the interface is connected using a single network PVC to a target link. The target link is usually attached to a remote interface, but it may be attached to the interface originating the PVC.

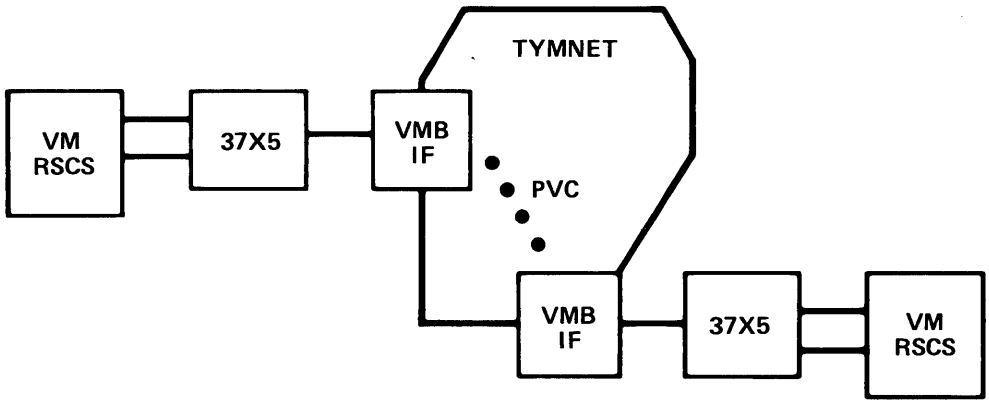


Figure 8-5. TYMNET VMB Support Application

Features of the VMB/RSCS interface are as follows:

- The interface supports a maximum of eight 9600 bps links using standard synchronous ports.

- The interface provides good batch throughput. Local acknowledgments insure continuous flow through the network.

- Data is transported reliably and transparently by the interface. End-to-end acknowledgment is used for the last block of every file being sent. This provides data integrity should a fatal error occur during file transmission. To meet the transparency requirement, the VMB interface conforms to protocol procedures defined in the IBM VM/370 RSCS Networking Logic Manual, LY 24 5203-1, and by actual VMB session analysis.

- With the exception of fatal errors, error recovery is handled locally. This is achieved by simulating logic of the remote VMB line driver in the local VMB interface. If data is lost due to a fatal error, the file must be retransmitted by the host.

TRANSPARENT BISYNCHRONOUS INTERFACE

The TYMNET Transparent Bisynchronous Interface provides a basic level Tymnet-based network service to host and terminal systems which communicate using non-IBM standard BSC protocols.

The interface supports a protocol requiring high reliability by transporting all BSC data frames end-to-end through the network. This protocol does not use standard BSC control frames for error recovery and flow control. Thus, all error recovery and flow control procedures occur end-to-end through the network, and a network failure cannot result in loss of data.

This product is best used in applications where low throughput and long delays are acceptable. The type of protocol best supported has the following characteristics:

- contention point-to-point protocol
- multiple frame acknowledgment
- full duplex
- batch-oriented traffic

Protocols that suffer significant degradation in throughput and response times using this interface product have the following characteristics:

- polled multipoint protocol
- only one frame acknowledgment
- half duplex
- interactive traffic

Figure 8-6 illustrates the application of the TYMNET Transparent Bisynchronous Interface to connect a host and terminal system. Each attached interface link is connected using a single network PVC to a target link. The target link is usually attached to a remote interface, but may be attached to the interface originating the PVC.

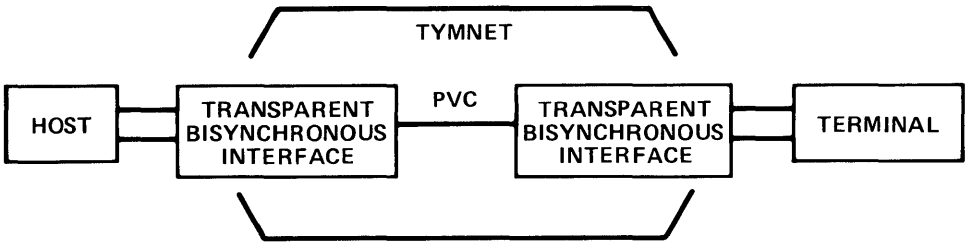


Figure 8-6. TYMNET Transparent Bisynchronous Support

Features of the TYMNET Transparent Bisynchronous Interface are as follows:

- The interface supports a maximum of sixteen 4800 bps or eight 9600 bps synchronous links using standard synchronous ports.
- The interface supports dial-up, leased, half-duplex, and full duplex modem facilities. An interface port with modem signaling is capable of DTR/DSR and RTS/CTS signaling.
- The interface provides a network connection without input from the attached host or terminal systems.
- Data is transmitted reliably and transparently by the interface. To insure reliable data transmission, all data and control frames are transmitted end-to-end. Timeout values in the host or terminal may require extension to compensate for network delay. For this implementation, the interface recognizes EBCDIC bisynchronous data frames as defined in IBM Binary Synchronous Communications, General Information, GA27-3004.
- Only data frames are transmitted end-to-end. Padding, control frames, and idle line conditions between frames are not transmitted.
- Unrecognized frames or frames with a bad CRC are discarded at the receiving interface.

3270 Interface

CHAPTER 9

3270 INTERFACE

CONTENTS

OVERVIEW

The three 3270 Bisynchronous (BSC) products for operation within the TYMNET network are the 3270 Host Interface, the 3270 Terminal Interface, and the Character Mode Translator.

Tymnet's special purpose operating system for communication processing, the Internally Switched Interface System (ISIS), provides a multiprogramming environment for the operation of several communications interface programs. Each 3270 interface runs in an ISIS slot in the same or in another TYMNET Engine.

The TYMNET 3270 Host Interface provides the 3270 terminal user access to a 3270 host computer through a TYMNET network. The host interface also provides users of certain ASCII CRT terminals access to 3270 host computers and supports ASCII printers that function as 3270 printers.

The TYMNET 3270 Terminal Interface provides access to host computers with 3270 support and access to interactive ASCII host computers through a TYMNET network. The 3270 terminal is one of the primary means for accessing IBM computer applications. It uses remote multidrop leased lines with polled BSC protocol.

The Character Mode Translator (CMT/3270) allows ASCII terminals to access 3270 hosts. A CMT communicates with a 3270 host interface using the 3270 Display System Protocol (3270 DSP).

All the capabilities of the BSC 3270 interfaces are available without hardware or software changes to the host computer. The host interface can appear as one or more 3270 BSC control units on a BSC line to the 3270 host. Both the host and the terminal interfaces communicate at speeds up to 9600 bits per second using IBM's 3270-Polled Bisynchronous Protocol. The 3270 terminal interface provides local control of terminal polling, device selection, and data block acknowledgement, which are all features of 3270 control units on multidrop lines. The configuration of the 3270 interfaces takes place when the system is generated.

Protocol

The five environments in which the 3270 interfaces operate are the following:

- native mode
- virtual terminal mode
- virtual host mode
- native mode with CMT/3270 emulator
- native mode through X.25 interface

The 3270 native mode protocol, which the interfaces use to communicate through the network, is compatible with the 3270 DSP. This enables users to communicate through an X.25 interface to other hosts and networks supporting the 3270 DSP protocol.

The X.25 allows for high speed line connections to the Front End Processor (FEP). As a result, FEP capacity is improved and therefore reduces further costs.

Early Data Forwarding

Early Data Forwarding allows the network interfaces to forward the bisynchronous blocks as they are being received rather than waiting for the entire block to be received. This has the effect of overlapping the serialization of the block at each step in the transfer from host to terminal.

Protocol Benefits

Tymnet is expanding the community of network users by adding support for industry standard protocols. Initial services are of a native mode extension cord capability which replaces long distance leased lines. The ability to select a destination on a per session basis allows access to a larger number of applications from a single terminal.

The creation of virtual modes also expands the number of users. The virtual terminal mode masks the special characteristics and provides access to ASCII terminal-oriented network applications. The virtual host mode allows some ASCII terminals access to specific industry standard protocol applications.

Native Mode

A 3270 terminal user operates in native mode when a 3270 terminal is communicating through network lines to a 3270 host. The interface function is user transparent except for the network services screen and menu screen. The network services screen provides a variety of interface control functions. The menu screen provides the terminal user with the ability to select a specific host or application.

TYMNET 3270 Host and Terminal Interfaces operating in the native mode replace leased lines between hosts using the 3270 protocol and compatible terminals. This compatibility allows for sharing of network facilities among multiple 3270 applications or for sharing 3270 applications and batch BSC or asynchronous applications.

Figure 9-1 illustrates a native mode connection.

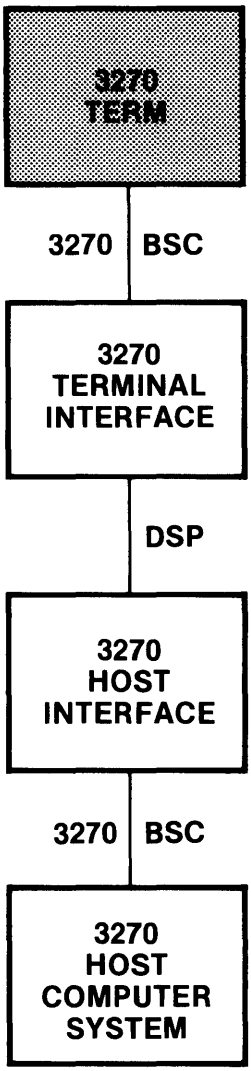


Figure 9-1. Native Mode

Virtual Terminal Mode

The user is in virtual terminal mode when the 3270 terminal communicates through the network to an interactive ASCII host. The 3270 character set is translated into ASCII and the interface performs keyboard and display formatting functions.

Virtual terminal mode expands 3270 terminal use and allows access to network applications that are oriented towards a simple ASCII terminal.

A 3270 terminal connected to the TYMNET 3270 Terminal Interface replaces a teletype-compatible terminal to access a network host computer through a TYMNET TYMCOM Interface.

Figure 9-2 illustrates a virtual terminal mode connection.

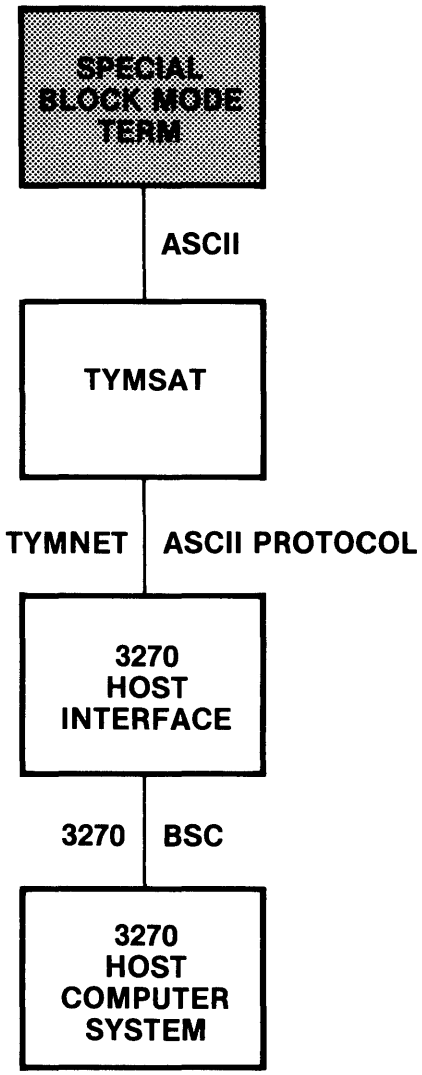


Figure 9-2. Virtual Terminal Mode

Virtual Host Mode

The virtual host mode makes it possible for certain ASCII block mode terminals to access 3270 protocol host computers through the TYMNET 3270 Host Interface. This mode increases the number of 3270 application users.

The host interface translates terminal input from ASCII to EBCDIC and formats the data into standard BSC messages. Output from the host computer is translated from EBCDIC to ASCII, and 3270 commands and orders are translated to screen and control sequences recognized by the terminal.

The translation process required is unique to each supported ASCII terminal model. Support is limited to the following terminals:

- IBM 3101
- Perkin-Elmer Owl 1200 series
- TYMSHARE 470

These terminals were selected for cursor control and field properties that minimize the amount of data for transfer through the network. Terminal selection was based on efficiency and response time.

The TYMNET 3270 Host Interface provides two methods of printer support in virtual host mode. The first method allows an ASCII printer to be logged in to a printer port on the 3270 host interface. In the second method, an ASCII printer is connected to the printer port on a Perkin-Elmer OWL terminal or a TYMSHARE 470 terminal. The second method requires that the host configuration associate the printer with the terminal. As a result, a terminal may be used to direct host output to the printer.

Figure 9-3 illustrates a virtual host mode connection.

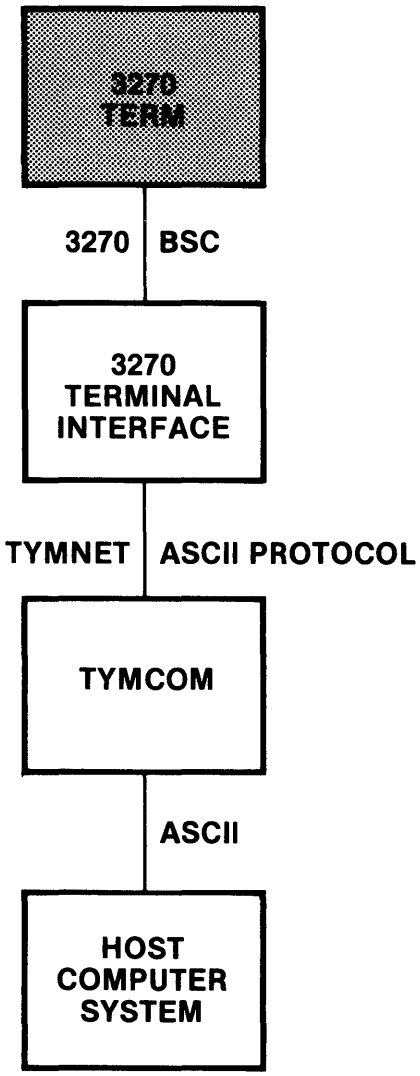


Figure 9-3. Virtual Host Mode

Native Mode with CMT/3270 Emulator

The CMT/3270 emulator supports terminals operating in ASCII character mode and allows less intelligent terminals to communicate with the TYMNET 3270 Host Interface operating in native mode. Because CMT/3270 uses DSP to communicate with the host interface, it is also possible to use the X.25 interface. The following are currently supported ASCII character mode terminals:

- ADDS Viewpoint 60/90
- ADDS Viewpoint 78
- ADM-3A
- ADM-11/1178
- Dasher D100/200
- Displayphone
- Hazeltine 1500 series
- Scanset
- Televideo 920/925
- VT-100

Since support for additional terminals is added regularly, a current list of supported terminals should be requested from a Tymnet representative.

The basic function of the CMT/3270 interface is to translate characteristics between a host with 3270 devices and certain ASCII terminals. The essential components of this function are an emulation of local 3270 keyboard functions and a translation of data formats. A secondary function is to reduce the character traffic to achieve the fastest response time and minimize network congestion.

Figure 9-4 illustrates the native mode connection options with the CMT/3270 emulator.

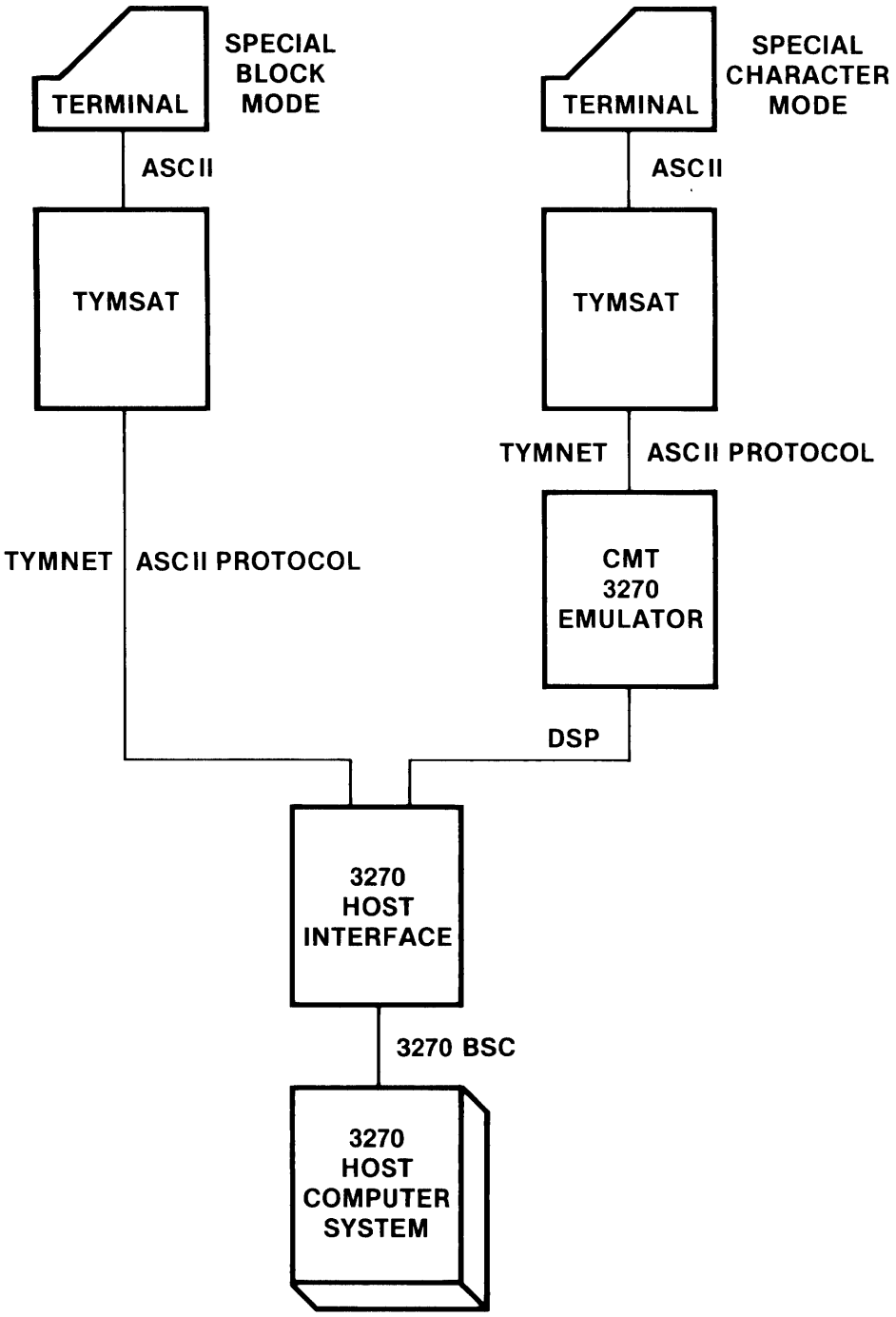


Figure 9-4. Native Mode with CMT/3270 Emulator

X.25 Interface

The 3270 DSP allows an X.25 network connection to provide access to 3270 applications. The 3270 DSP is a fourth layer of protocol above the three X.25 protocol layers and is used to transfer 3270 information.

A Packet Assembler/Disassembler (PAD) can substitute for the host or terminal interface. The PAD communicates with a device by a BSC line or by a more direct mechanism such as a host channel attachment.

Figure 9-5 illustrates the various X.25 connection options.

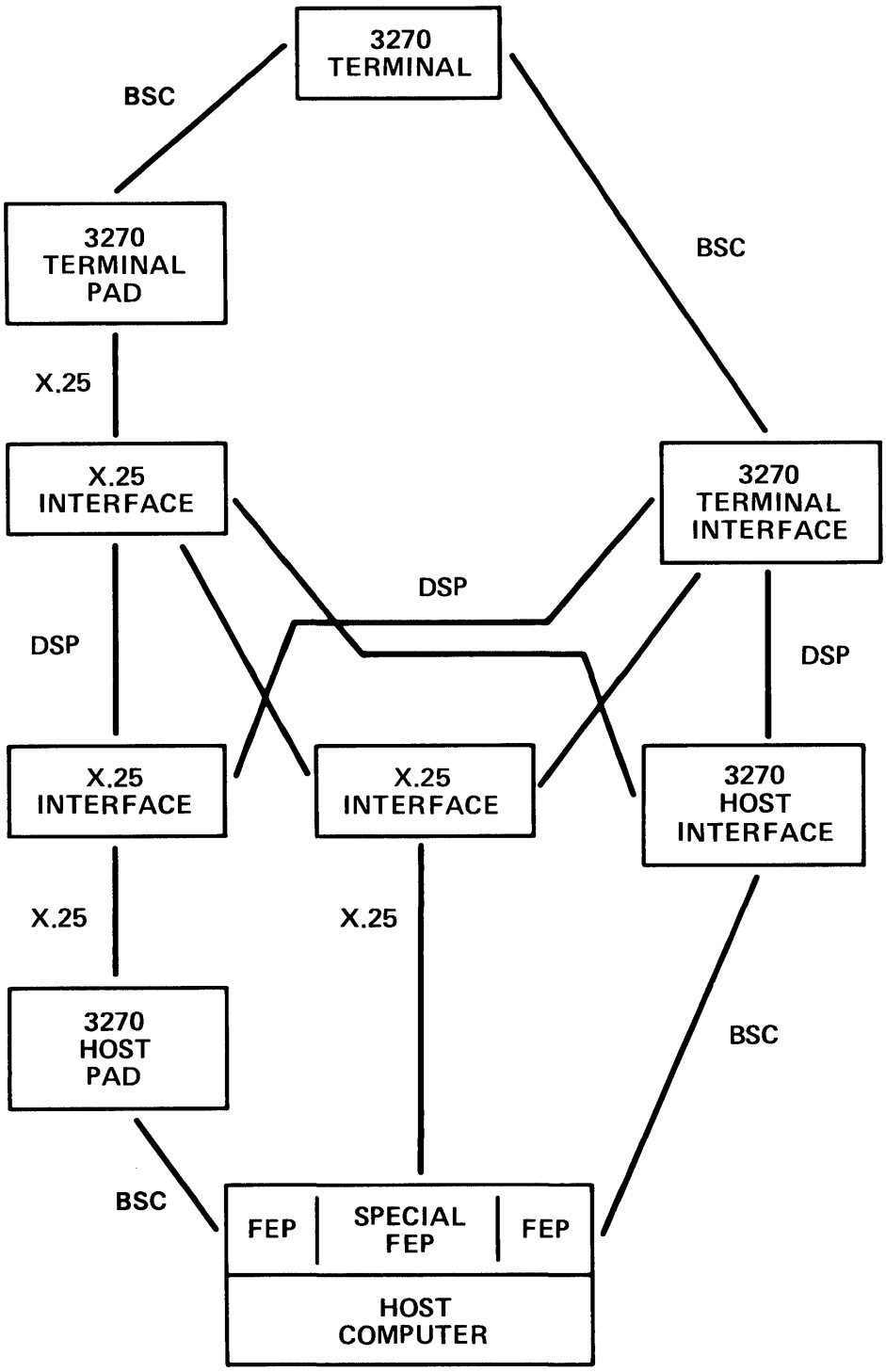


Figure 9-5. X.25 Interface

Support Capabilities

The TYMNET 3270 Interfaces are compatible with 3270 terminal equipment and equivalent devices not manufactured by IBM. These devices are compatible with the 3270 polled BSC protocol. Combined with virtual circuit switching, protocol decoupling, and terminal emulation capabilities, Tymnet's implementation of 3270 support eliminates both geographical and functional dependencies between terminals and hosts. Consult a Tymnet representative to verify the compatibility of specific 3270 devices not manufactured by IBM. Refer to Table 9-1 for TYMNET 3270 Interface support capabilities.

Table 9-1. Compatible IBM Devices

Number	Type
3178	Display terminal
3271-1,-2	Control unit
3274-1C	Control unit
3276-1,-2,-3,-4	Control unit/display
3277-1,-2	Display terminal
3278-1,-2,-3,-4	Display terminal
3279-2,-3	Display terminal
3284-1,-2,-3	Printer
3286-1,-2	Printer
3287-1,-2	Printer
3288-2	Printer
3289	Printer
3290	Information panel

Uppercase and lowercase alphabetic characters are fully supported.

SDLC

To be supplied at a later date

SNA

CHAPTER 11

SNA INTERFACE

CONTENTS

OVERVIEW

This is a twofold document: its main purpose is to provide an overview of the TYMNET SNA Interface, and secondarily this document provides an overview of IBM's System Network Architecture (SNA), its history and purpose. Readers familiar with SNA concepts are invited to skip the background information provided in the next section and to begin at the section titled "The TYMNET SNA Interface."

IBM System Network Architecture

IBM designed SNA to enable customers to construct integrated networks. Before SNA was developed, IBM had several hundred communication products, using dozens of teleprocessing access methods with more than a dozen data-link protocols. Given the desire of many of IBM's customers for compatibility between these mutually incompatible programs and protocols, IBM created a network architecture that would provide a coherent framework for loosely coupled, distributed data processing.

Nodes

An SNA network comprises four types of machines, called nodes:

1. Type 1 nodes are terminals.
2. Type 2 nodes are cluster controllers, machines that supervise the behavior of terminals and other peripherals.
3. Type 4 nodes are front-end or remote communication controllers, devices that relieve the host CPU of the network and interrupt handling associated with data communication.
4. Type 5 nodes are mainframe hosts.

NOTE

There are no type 3 nodes.

The following figure shows a simplified IBM SNA network.

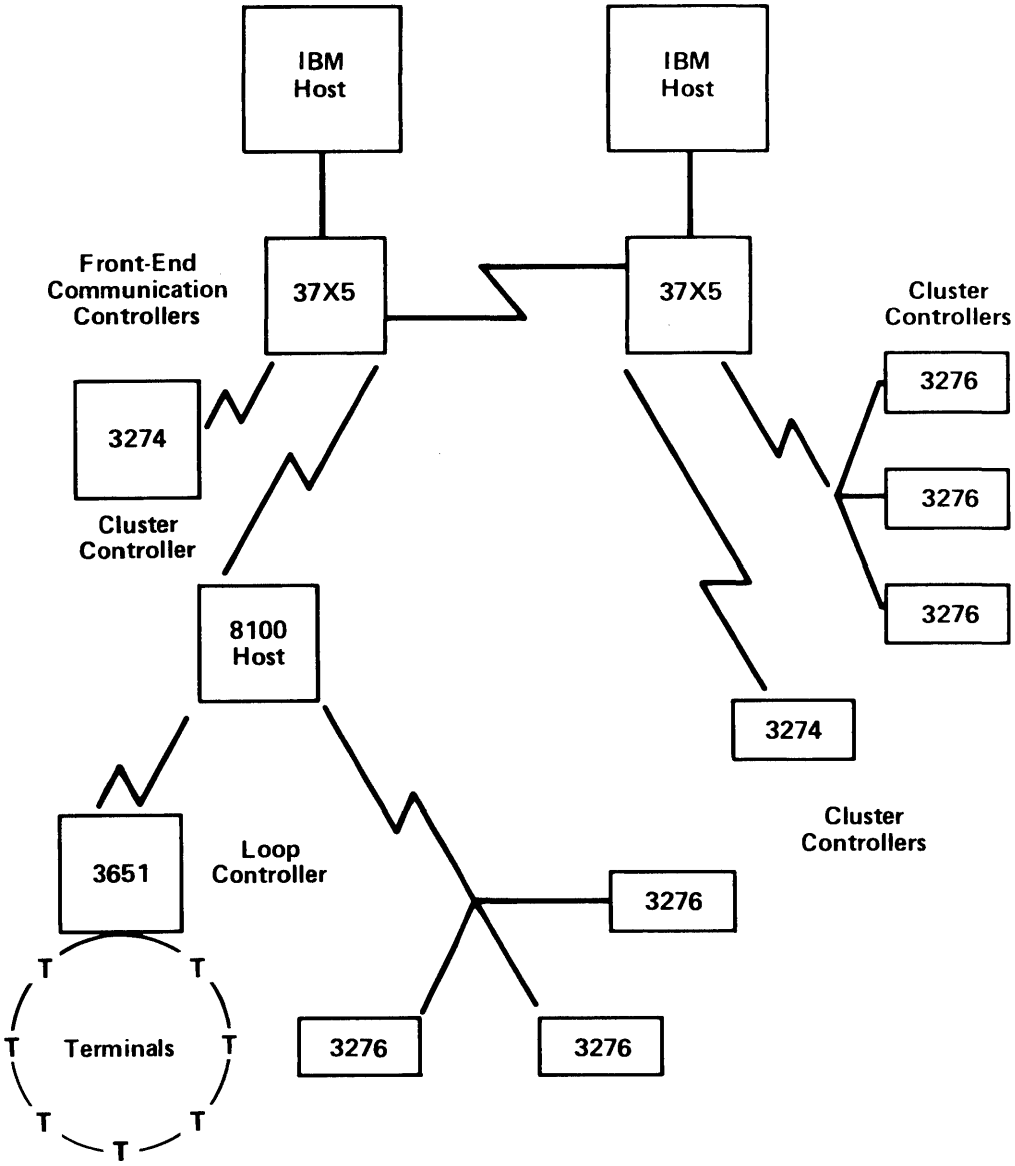


Figure 11-1. A Simple SNA Network

Each node contains one or more network addressable units (NAUs). There are three kinds of NAUs: a logical unit, a physical unit, and a system service control point.

A logical unit (LU) is a bridge between a user or an application program and the SNA network. Through its LU, a user application accesses network resources, sends data into the network, and receives data from the network.

A physical unit (PU) is an NAU, associated with each node, which is used by the network to bring the node online, take it offline, test it, and perform similar administrative functions. The PU provides a way for the network to address a physical device, without reference to which processes are using it.

The third kind of NAU is the system service control point (SSCP), which is only used on one host mainframe (PU type 5) per SNA network. The SSCP acts as a supervisor. It has complete knowledge and control of all the front ends, controllers, and terminals attached to the host on which the SSCP resides. The collection of hardware and software managed by an SSCP is called a domain. Figure 11-1 depicts a two-domain SNA network.

Hierarchical Structure

A key SNA concept is the division of the communication system functions into well-defined logical layers (see Figure 11-2). SNA is structured in layers for two reasons: to permit changes to be made in one layer without affecting other layers and to allow interactions between functionally paired layers in different units. (This pairing is required to support distribution of function.) The SNA protocol hierarchy consists of the following logical layers:

1. Physical link control layer. This layer takes care of physically transporting bits from one machine to another.

2. Data link control layer. This layer constructs frames from the raw bit stream, detecting and recovering from transmission errors in a way transparent to higher layers. Many networks have directly or indirectly copied their layer 2 protocol from SNA's layer 2 data communication protocol, SDLC.

3. Path control layer. This layer is concerned with routing and congestion control within the network. It can block unrelated packets together into frames to enhance transmission efficiency and can deal with the hierarchical addressing used in SNA.

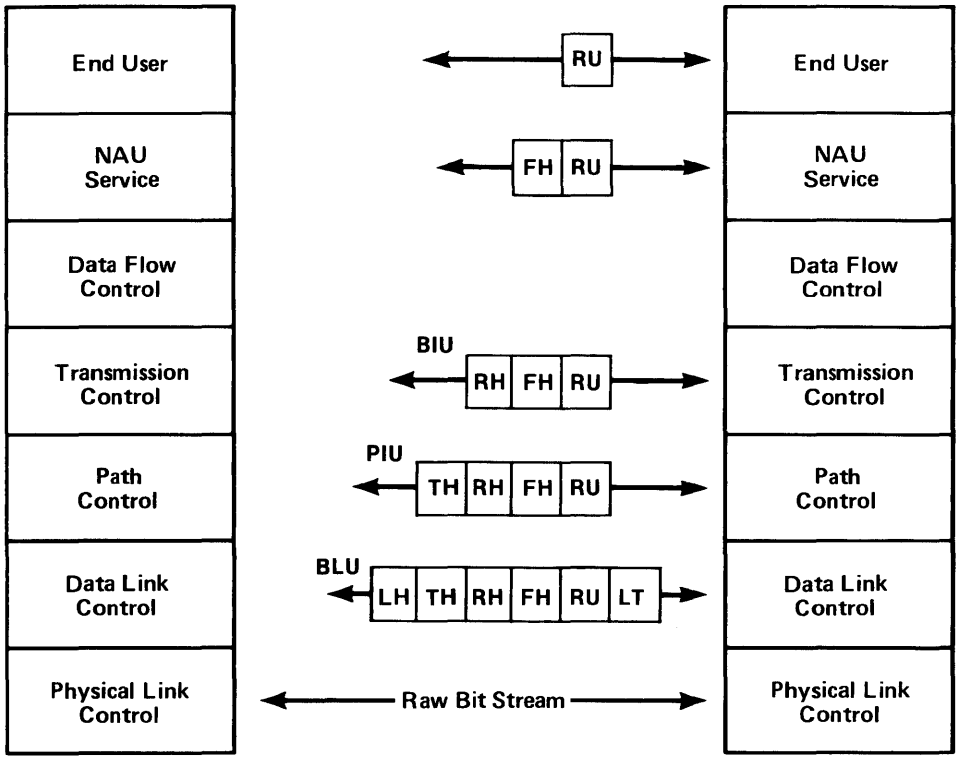
4. Transmission control layer. This layer's job is to create, manage, and delete transport connections, called sessions in SNA. In effect, this layer provides a uniform interface to higher layers, independent of the properties of the network. Once a session has been established, the transmission control layer provides the following functions:

   - regulation of the rate of flow between processes
   - control of buffer allocation
   - management of the different message priorities
   - multiplexing and demultiplexing of data and control messages for the benefit of higher layers
   - encryption and decryption when requested

5. Data-flow control layer. This layer keeps track of which end of a session is supposed to talk next, for processes that need such a service. This layer is also heavily involved in error recovery. An unusual feature of the data-flow control layer is the absence of a header used to communicate with the corresponding software on the other end. Instead, the information normally communicated in a header is passed to the transmission control layer as parameters and is included in the transmission header.

6. Network-addressable units services layer. This layer provides two classes of service to user processes. First, there are presentation services, such as text compression. Second, there are session services

for setting up connections. In addition, there are network services that have to do with the operation of the network as a whole.



RU = Request/Response Unit
FH = Function Header
RH = Request/Response Header
TH = Transmission Header
LH = Link Header
LT = Link Trailer

BIU = Basic Information Unit (= message)
PIU = Path Information Unit (= packet)
BLU = Basic Link Unit (= frame)

Figure 11-2. Protocol Hierarchy and Units Exchanged in SNA

The TYMNET SNA Interface

The TYMNET SNA Interface is a software product that runs in a slot on a TYMNET node under the Internally Switched Interface System (ISIS). The SNA interface provides access between SNA-protocol terminals and hosts through the TYMNET network. The SNA interface translates SNA communication protocols into the synchronous packet-switching protocol of the TYMNET network and translates the TYMNET communication protocol back into the protocol that the SNA nodes can process.

SNA communication interfacing is required at both the SNA terminal and SNA host ends of the network, and both the host and terminal interfaces can be provided by TYMNET in the same software package.

Figure 11-3 shows the SNA interface at both the host and terminal ends of the network.

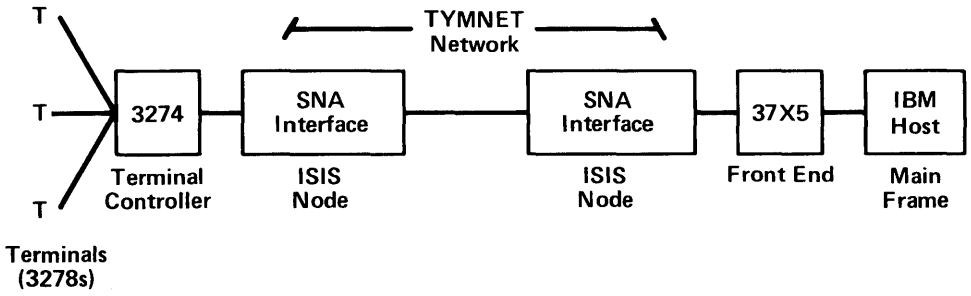


Figure 11-3. TYMNET SNA Interface

The SNA host interface connects the host front-end communication controller with TYMNET. This interface appears as one or more cluster controllers to the host front end.

The terminal interface connects the SNA cluster controllers with TYMNET. The terminal interface contains a system service control point (SSCP), which provides local control of the SNA cluster controller. Thus, when connected to the TYMNET SNA Interface through an SNA cluster controller, SNA terminals can individually access different IBM hosts and applications, without using host-based routing.

The TYMNET SNA Interface preserves all of the original capabilities of the communication controller and remote cluster controller, with little or no change to the system generation of the controllers.

Figure 11-4 illustrates the interconnection of SNA/SDLC hosts and terminals using TYMNET.

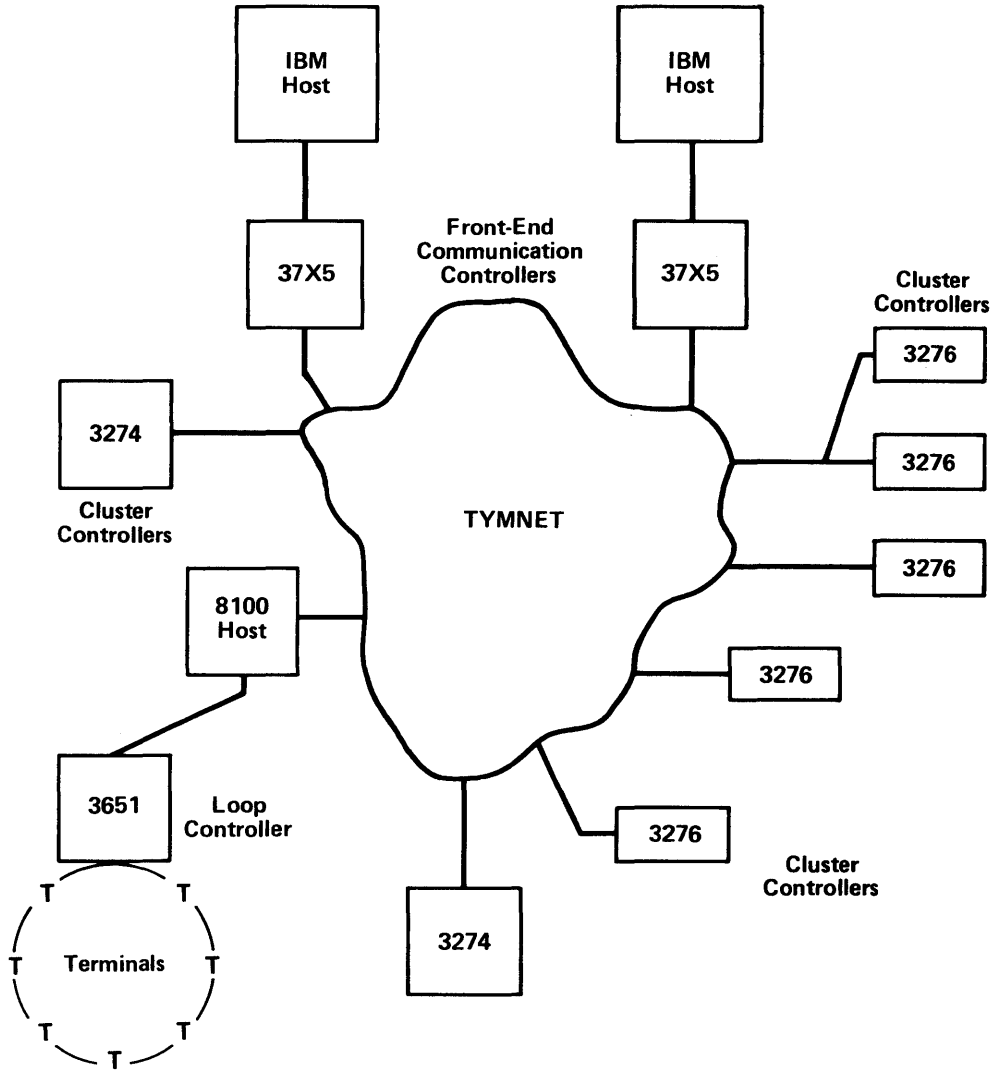


Figure 11-4. TYMNET/SNA Network

Support Capabilities

The TYMNET SNA Interface supports both Normal Disconnect Mode (NDM) and Normal Response Mode (NRM).

The host interface supports point-to-point or multipoint, nonswitched, half-duplex links to hosts. A "secondary" (host) port will appear to the host computer as a single secondary station on a point-to-point link or as a number of secondary stations on a multipoint link.

The terminal interface supports point-to-point or multipoint, nonswitched, half-duplex links to terminals. A "primary" (terminal) port will appear to attached terminals (on a point-to-point or multipoint link) as a primary station.

The TYMNET SNA Interface supports physical unit (PU) types 2 and 4 and logical unit (LU) type 2.

Additional support capabilities are provided by the SNA DSP Virtual Host.

The SNA DSP Virtual Host

The purpose of the TYMNET SNA DSP Virtual Host is to translate Display System Protocol (DSP) into SNA LU type 2 protocol. This product allows either bisynchronous 3270 terminals attached to a 3270 terminal interface or ASCII asynchronous terminals logged on to the Character Mode Translator (CMT) interface to access IBM SNA hosts as LU type 2 devices.

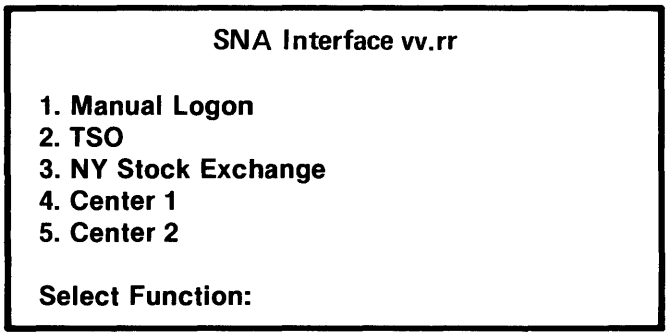
Because inactive terminals must be logged off the SNA host after a specified time period, the network must complete a sign-off sequence to the SNA host. To accomplish this, the TYMNET SNA DSP Virtual Host has an inactivity timeout. The interface issues an application and network logoff when this timeout expires. The timeout value can vary and is defined in the system generation file for the interface. Also, the timeout and sign-off option can be enabled or disabled for each device.

The Virtual Host supports DSP Connect Request Modes (CRMs) currently available through TYMNET's CMT and Bisynchronous 3270 Interface products. The CRMs 1, 2, and 3 are supported for native mode only. For virtual mode, only CRM 3 is supported. The DSP menu provides several fields for specifying a desired LU. Nonspecific requests are routed to any available LU on the interface.

Operation

The TYMNET SNA Interface provides the SNA terminal user with a variety of network services through the Network Services Manager (NSM). The NSM provides these services (such as network logon) through formatted display screens. The following user-friendly screens are available:

1. The Network Services Menu Screen allows the user to choose between manual logon and preset logon options. The user selects a menu entry by pressing the key corresponding to the entry. The following is an example of a Network Services Menu:



2. The Manual Logon Screen guides the user to enter proper network logon information. This screen enables the user to choose from any available destination host.

3. The Display System Protocol Screen asks the user to provide information for attaching the terminal LU to the desired LU on the host.

These screens are defined in the SNA interface system-generation file (called the Tymfile). A set of screens is assigned to each LU (in this case, each terminal port). Thus, two LUs on the same controller can be assigned two different screen sets.

Configuration

The system generation of the SNA interface is guided by the configuration file called the Tymfile. The Tymfile contains individualized macro statements that generate ISIS slot code tailored to the specific configuration. These macro statements are divided into seven categories:

1. Control directives specify whether or not a configuration parameter listing will print out and what the listing will contain. The configuration parameter listing displays, at the user's ASCII terminal, the resulting functions and characteristics of an SNA interface system generation. Thus, the user can check the Tymfile against the actual configuration and make any desired changes.

2. Network and ISIS statements configure the SNA interface to conform to the internal rules of TYMNET and ISIS. The ISIS operating system contains rules on how interfaces are constructed and how they communicate with each other.

3. Protocol group statements are the first level in a macro hierarchy that specifies SNA interface characteristics. These macros are called SNA configuration aggregate macros. As with other aspects of SNA, these macros are designed in distinct layers. In fact, the SNA interface can be viewed as a hierarchical arrangement of elements, with each element type represented by a macro type.

   One advantage of this hierarchical structure is economy in coding effort, because macros at lower levels can specify exceptions to higher-level macros.

   The highest-level SNA configuration aggregate macros, protocol group statements, configure link-level characteristics of the entire interface. These characteristics include line speed, transmission mode, window size, and even specific characteristics of PUs and LUs. Exceptions can be specified later in the Tymfile.

4. Line statements specify line characteristics for the SNA interface. Each line in the SNA interface is represented by a line aggregate macro, whether the line is non-

switched multipoint, nonswitched point-to-point, or switched point-to-point. Line statements are the second highest level in the hierarchy of SNA configuration aggregate macros and can specify exceptions to the protocol group statements.

5. PU statements specify the physical-unit characteristics of an SNA node, such as a terminal, controller, or host front-end.

6. LU statements specify the logical-unit characteristics of an SNA LU device.

7. Network service (NS) statements define the content and functions of the Network Services Menu Screen. Network logins can be predefined with NS statements.

PROBE

CHAPTER 12

PROBE

CONTENTS

OVERVIEW

PROBE is an interactive network-monitoring tool that runs on every Supervisor in the TYMNET network. It is used to alter the status of network components, as well as to monitor network status. PROBE is used by the network operations and technical support personnel responsible for overall network maintenance. PROBE runs as a Supervisor slave subprogram on a TYMNET ISIS-II (Internally-Switched Interface System) Engine that is running the Supervisor program in one of its ISIS-II job slots. PROBE and five other slave subprograms run in job slots on this Engine (as shown in Figure 12-1). PROBE has access to the Supervisor and the other slave subprograms through shared memory.

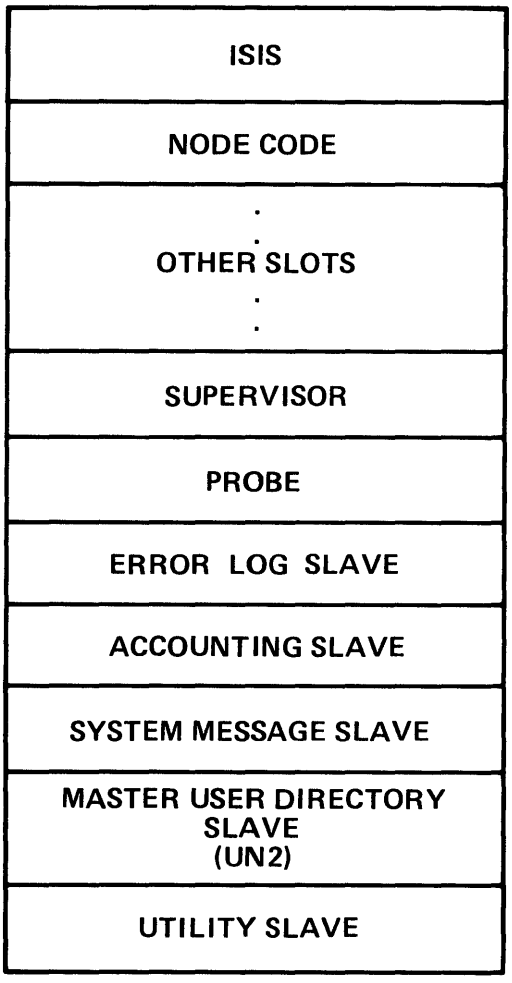


Figure 12-1. PROBE on the Supervisor Node

Using PROBE, an approved class of users may monitor the network as follows:

- determine network usage
- determine network component status
- stop use of communication links with high error rates
- route new circuits over alternate paths

PROBE is able to monitor the network because it has access to a current picture of the network topology. It obtains the network topology through the following means:

- accessing node descriptor and host tables stored in the Supervisor's memory
- accessing the network Error Log file stored on disk
- using its circuit-tracing facility to instruct the Supervisor to request circuit information from the origination or destination nodes of a circuit

PROBE is always in a position to access the Supervisor without depending on any external communication links. While a network may contain as many as seven Supervisors, only one Supervisor is active (awake) at a time. On the active Supervisor, PROBE is used to monitor or alter the current status of the network. On an inactive (sleeping or frozen) Supervisor, PROBE is used to obtain log information generated while that Supervisor was awake and to determine the status of the network when the Supervisor went to sleep or was halted.

PROBE User Status

Because PROBE is a powerful tool for Network Operations Center and maintenance personnel, access to the system commands is restricted to certain classes of users as shown in Table 12-1.

Table 12-1. PROBE Status Classification

Status	Function	Users
none	read Supervisor Log; obtain general network information	all PROBE operators
20	obtain circuit status information and shut links	network operators system administrators system programmers
10	alter status of nodes	network operators system administrators system programmers
01	alter status of Supervisor	network operators system administrators system programmers
02	perform network maintenance	network operators system administrators

Supervisor Log Messages

The active Supervisor collects and stores chronologically the log messages generated by itself, by network nodes, and by network hosts. These log messages document conditions and events reported by the Supervisor, by node code running in any of the nodes, or by host processes. The log messages provide information about the following:

- network activity and performance
- network nodes
- network hosts
- network links
- operator interventions

PROBE commands may be used to read the log messages. On the active Supervisor, PROBE commands may be used to monitor current network conditions by reading log messages as they are generated. However, PROBE commands may also be used on all Supervisors (active or inactive) to obtain a history of network conditions by reading previously generated log messages.

Particular network components or conditions may be monitored by using PROBE commands to restrict the log message display in the following ways:

- The messages may be read beginning at a specified time.
- The messages may be screened so that only certain types of messages are displayed. For example, they may be screened to display line errors in the entire network.
- The messages may be screened to display only information about one or two nodes.
- The messages may be screened to display information about a specific interface or product in the network. For example, they may be screened to monitor all host status changes.

Network Status Information

In addition to monitoring the status of the network by reading the Supervisor log, PROBE has several commands that are used to obtain specific current information about the network, as follows:

- the version of code and the hosts running on a node and the links connected to that node
- network performance and usage
- Supervisor performance
- link status
- circuit status
- port status

PROBE commands may also be combined to display information about a particular combination of events and/or network components. For example, they may be screened to display line errors reported by a specific node in the network.

Network Control

PROBE commands may be used to alter the status of one or more network components when a problem affecting network performance is detected, as follows:

- To shut a link that requires maintenance or is overloaded. Shutting a link prevents the Supervisor from building additional circuits using that link. When the problem is resolved, the shut condition may be removed.
- To load a node with new code by activating its downline loading routine.

The commands that affect these network components have a direct affect on the topology of the network.

Supervisor Control

The highest level of PROBE commands are powerful because they affect the operation of network Supervisors. They are used to perform the following:

- validate PROBE users
- put one Supervisor to sleep and subsequently wake another
- take a Supervisor out of the network (halt and freeze the Supervisor) for debugging purposes and subsequently bring it back into the network
- modify the time of day in the Supervisor's Superclock when necessary
- perform maintenance operations on a Supervisor such as initialize the Supervisor's accounting file or error log

CONTENTS

OVERVIEW

The Tymnet Monitoring and Control System (TMCS) is a network monitoring tool and provides multiple users selective access to the status of a TYMNET network. Access can be limited to a specific network subset or limited to a subset of network events. Both current and past events are displayed.

As an interactive system, TMCS can be used to monitor the following three aspects of network status:

- The user can log in directly to PROBE through TMCS to monitor the current network status.
- The user can obtain a comprehensive listing of all Supervisor log messages to trace recent history of network status.
- The user can obtain a display of current network exceptions such as nodes, hosts, or lines, that are down, shut, or missing.

TMCS does not require direct Supervisor resources. Only one circuit to PROBE exists between TMCS and an active Supervisor. Therefore, users logging in to TMCS do not compete with the Supervisor for resources. TMCS automatically tracks the active Supervisor during a Supervisor change. The user does not need to log in again to the newly activated Supervisor.

TMCS Commands

The user profile determines access to network information. TMCS has six command groups. Users are authorized by the profile as to which commands to use. Table 13-1 lists each command group.

Table 13-1. TMCS Command Groups

Command	Definition
Network Monitoring	monitors the network
Remote Greeting Message	sets or clears remote greeting messages in TYMSATS
Profile Maintenance	creates, maintains, and deletes user profiles
Profile Modification	modifies a user's profile
Utility	performs basic functions
Program Maintenance	monitors and maintains TMCS itself

TMCS Log Messages

Log messages displayed using the TMCS Log command or the PROBE Log command are oriented to the operations environment. However, some messages indicate system problems; therefore, the operator should report them to a technical support representative. Log messages include information about the following:

- network nodes
- network lines
- network hosts
- network links
- node report types
- Supervisor observations

Report Criteria

Different types of test criteria are used to screen TMCS log messages for specific nodes, hosts, lines, links, and ports. They are listed as "report criteria" to indicate the requirements for generating output of special log messages. Test criteria are listed in Table 13-2.

Table 13-2. Log Message Test Criteria

Test	Application
Node	single node number
Host	single host number
Node/Host	host number and the node on which the host resides
Link	two neighbor nodes
Port	node number and node port number
Link/Port	two neighbor nodes and one connecting node port
Line	two neighbor nodes and each connecting node port

Alarm Facility

The primary function of the TMCS Alarm Facility is to use current TMCS information to trigger an alarm when a network fault (crash or outage) occurs. The TMCS network state description database facility has been enhanced to monitor new network operational conditions for the Alarm facility. In addition, each TMCS user is able to monitor local and central network faults by using alarms selection criteria.

The TMCS Alarm Facility has the following components:

- a user process for executing alarm code
- a series of commands to establish and manipulate variables with TMCS system-wide impact
- a series of user-specific commands

The TMCS Alarm Facility is dependent on the following two distinct categories of network conditions:

1. predetermined conditions that are identifiable by a start-and-stop sequence of events

2. threshold conditions that are identifiable by a recurring nature

The predetermined conditions are a collection of specific event categories that pertain to all TMCS users. These conditions are TMCS processes and are not changeable by any users.

The threshold conditions are a collection of events that are changeable by operations personnel. These conditions are categorized into special interest groups for the network controller.

The external alarm system comprises an alarm panel connected to a TYMNET Engine by an asynchronous port. If the alarm panel does not receive a periodic message, then the panel lights turn on.

Automatic Recovery and Reload Facility

The Automatic Recovery and Reload (ARR) is a single facility that detects failures and initiates recovery actions. It determines accurately if a node or host is down, and initiates and monitors a node or host core dump, restart, or reload.

Special elements must be known prior to an attempt at reloading for the ARR facility to function properly. Those elements are network architectural designations assigned to a TYMNET Engine and executable applications. The primary designation is a network node number assigned to the Engine itself. Additional elements of information concern the following:

- hosts that operate on the node
- host location (slot) within the node
- applications running on the hosts
- ISIS kernel host number
- TYMNET neighbor nodes interconnecting a node to the remaining network and Supervisor

At the heart of the ARR facility are two lists which contain available hosts and nodes for loading and those currently being loaded by TMCS (either automatically or interactively). The list of those nodes and hosts available for loading is maintained by network operations personnel. If the host or node is not activated in this list, then loading does not occur from within TMCS.

The list of hosts and nodes currently being loaded is maintained by the reload facility and prevents any hosts or nodes which are selected by multiple users from being repetitively loaded by each user.

NETVAL

CHAPTER 14

NETVAL

CONTENTS

OVERVIEW

Network Validations (NETVAL) is a user validation program that runs in an ISIS (Internally Switched Interface System) job slot in a TYMNET Engine. Using NETVAL, multiple users can simultaneously update and maintain the validation data that controls user access to the network.

Users can control network access by using NETVAL for the following:

- assigning login usernames and passwords
- assigning control options
- maintaining access profile information, which determines a user's extent of access into the network
- setting up tables of information used by the Supervisors to validate user login attempts

NETVAL users may add, change, or delete user validation data in the Controlling User Directory (CUD). The CUD is a collection on disk of all user validation data. When the NETVAL user updates the CUD, NETVAL also records the update data in the Master User Directory (MUD) Update file. The MUD Update process copies the data from this file to the MUDs, which are maintained on each Supervisor disk in the network. The active Supervisor handles requests for virtual circuits (logins) by referencing user validation data stored in the MUD. Figure 14-1 illustrates the NETVAL process.

NETVAL users may perform various operations based on the authorization levels and/or the control options assigned to their usernames. Some users (password-only users) may use NETVAL only to change their own password; other users (validators) may be assigned a validation authorization level. Users may also be assigned control options that allow them to perform additional operations regardless of their authorization level.

NETVAL has an operator log which contains information about all NETVAL logins and logouts. Data on specific user activity and on NETVAL events, including error messages, are available in this log. Certain NETVAL users may display the information from this log.

The NETVAL archive subsystem is used to back up NETVAL disk files to tape and to perform a number of magnetic tape operations.

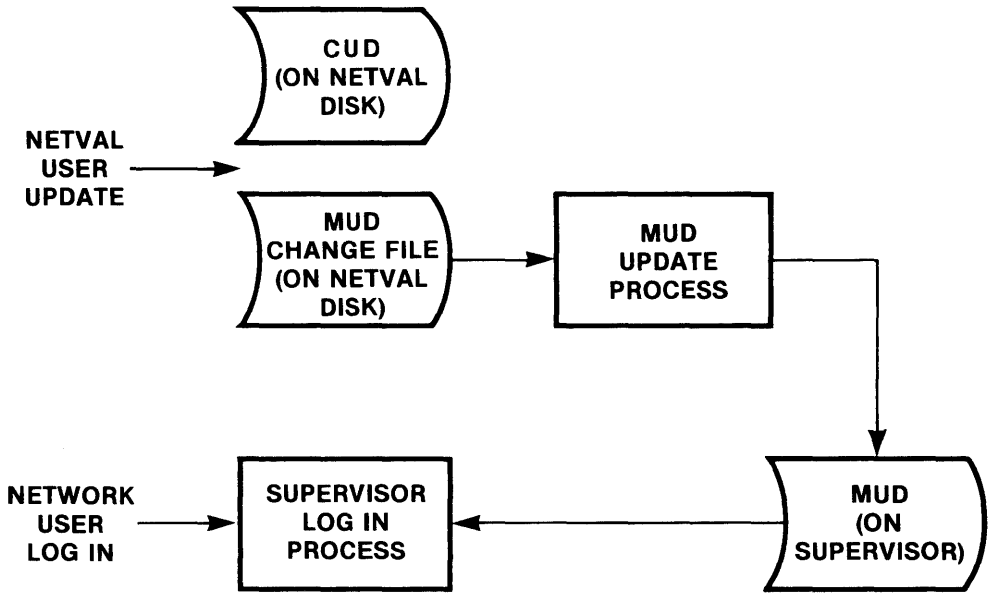


Figure 14-1. NETVAL Process

NETVAL TABLES

NETVAL maintains data that Supervisors use to control network access. Information about user origins (entry hosts and/or nodes from which a user may originate circuits) and destinations (hosts to which the user may build a circuit) is stored in four tables on the NETVAL disk. Table 14-1 defines these tables.

Table 14-1. NETVAL Tables

Table	Contents
Host List	host list names lists of host numbers
Node List	node list names lists of node numbers
Class	host and node list names (origins)
Group	host list names (destinations)

NETVAL uses data from these four tables to create the Supervisor Class and Group table, which the Supervisor references in verifying network logins.

CUD/MUD ENTRY DEFINITIONS

User validation data in the CUD consists of CUD entries. MUD entries are identical to CUD entries. A CUD/MUD entry includes a username and a Universal User Number (UUN), which identify a network user. In each entry, the origins and destinations that can be accessed with the username are defined in the access profile. The CUD/MUD entry can also contain options that limit or expand a user's extent of network access.

The following user entry fields are listed in the order in which they appear in a CUD/MUD entry.

Username

A user must have a valid username to log in to the network. One user may have more than one username.

Password Cipher

A username is assigned a password unless the No Password option is included in the user entry. The password is enciphered, and the 32-bit cipher is stored in the CUD/MUD entry.

User Entry Change Date

This field contains the date and Greenwich Mean Time (GMT) when any item in the user entry (other than the password) was last changed.

Global Account Number (Optional)

Global accounts can be used to group usernames for special purposes. When Global Account Numbers (GANs) are included in a network, all usernames are assigned a GAN. The decision to include GANS in a CUD/MUD entry is made during network configuration.

Universal User Number

Every username in a network is assigned a UUN, which uniquely identifies the username. UUNs may be used for accounting purposes.

Control Options

Control options may be assigned to a username when a user entry is created or modified. Ignore Host, No Password, Shut Override, and Transparent Login are four options that affect a user's network access. The remaining control options determine the operations that a NETVAL user can perform. These control options are Auditor, Operator, Account Supervisor, and Network Administrator.

No Password — When the No Password option is assigned, the user logs in to the network without using a password. Only users with passwords can use the colon (:) to specify a destination host. If the No Password option is assigned, the Ignore Host option is also automatically assigned to a username.

Ignore Host — When the Ignore Host option is assigned, the network user is not required to specify a destination by entering a colon (:) and a host number in the login string. The destination host is selected from the username's access profile, and the user is routed through the least-cost path to the host.

When either the No Password option or the Transparent Login option is assigned to a username, the Ignore Host option is automatically assigned. However, the Ignore Host option can also be assigned independently.

Transparent Login — When the Transparent Login option is assigned, the user logs in to a local network and passes through a gateway into a second TYMNET network without logging in again. In the initial network, the Ignore Host option is automatically assigned to users with Transparent Login.

Shut Override	When the Shut Override option is assigned, the network user can access a shut host. A shut host is one that is up but is not available for normal access.
Auditor	The Auditor option permits the NETVAL account supervisor validator or password-only user to display CUD entry information for any username.
Operator	The Operator option permits the NETVAL account supervisor validator or password-only user to use the archive subsystem and to perform NETVAL maintenance operations.
Account Supervisor	The Account Supervisor option defines an authorization level that enables the NETVAL validator to display, add, modify, or delete information for usernames in the validator's GAN.
Network Administrator	The Network Administrator option defines an authorization level that enables the NETVAL validator to display, add, modify, or delete information for any username in the CUD. This authorization level also enables the validator to maintain origin and destination data in the NETVAL and Supervisor tables.

District Number (Optional) When the district number is included in a network, all usernames are assigned a district number. The decision to include this number in a CUD/MUD entry is made during network configuration. Districts can be used to group usernames for special purposes.

Password Change Date This field contains the GMT date when the username's password was last changed.

Access Profile

The remaining data in the user entry is access profile information. The access profile entries are pairs of origins and destinations that are assigned to the username.

The origins may be defined in the following two ways:

username-specific list	One or more originating host and/or node numbers.
class	A globally defined list of origins identified by a class number. A class numbered 1 or greater is made up of names of host lists (lists of host numbers) and node lists (lists of node numbers). Class 0 is the universal class. It allows access from all origins in the network.

Classes and username-specific origin lists can be positive or negative. A positive class or username-specific list includes only the origins specified. A negative, or exception, class or username-specific list includes all network origins except those specified.

Destinations may be defined in the following two ways:

username-specific list	One or more destination host numbers.
group	A globally defined list of destinations identified by a group number. A group numbered 1 or greater is made up of host list names. Group 0 is the universal group. It allows access to all destinations in the network.

A group can be positive or negative. A positive group includes only the destinations specified in its definition, whereas a negative (exception) group includes all network destinations except those specified.

A positive group or a host may be specified as a default destination (home). However, Group 0 or a negative group may not be specified as homes. Users who are assigned the No Password, Ignore Host, or Transparent Login options have all the destinations in their access profile entries defined as homes. However, other users are not required to have any destinations defined as homes in their access profile entries.

SUPERVISOR CAPABILITIES

Login Verification

The active Supervisor verifies requests from users for network access by referencing validation data supplied by NETVAL. The Supervisor checks the MUD for an entry that matches the username and password entered in the login string. If a match is found, the Supervisor compares the rest of the login information to the corresponding MUD entry.

The active Supervisor also uses MUD data, supplemented with information from its class and group table, to determine a user's extent of network access. When a network user attempts to initiate a circuit from an originating node or host, the Supervisor checks the username access profile for this origin. One or more classes may be listed in the access profile of the user requesting a circuit. If this is the case, the Supervisor checks its class table for originating node and/or host numbers that are included in each one of these classes.

When the user specifies a destination host number, the Supervisor checks for a corresponding origin/destination pair in the user's access profile. One or more groups may be listed in the user's access profile. If so, the Supervisor checks its group table for the destination host numbers in each of these groups. When the Supervisor has verified that the requested origin and destination can be accessed by the user, the circuit is built.

When no destination is specified by the user, the Supervisor checks the user's access profile to determine whether a default destination is associated with the user's origin. A positive group or a destination host can be defined as a home. When a user has a home specified in the access profile, the user does not need to enter a destination host at login but instead will be routed automatically to his home. Users with the Ignore Host option are not required to specify a network destination in the login string. In this case, a destination host is selected from the username access profile, and the user is routed through the least-cost path to that host.

The following NETVAL processes update or verify the information used by the Supervisors for validations and circuit building.

MUD Updating

When a change is made to the CUD, the description of this change is appended to the MUD Update file on the NETVAL disk. An automatic process identifies data in the MUD Update file that needs to be sent to a Supervisor to update its MUD. A separate process exists for each Supervisor; all of these processes operate simultaneously. Change data that is sent to a Supervisor is written to the MUD on the Supervisor disk. Each Supervisor's MUD is continuously updated to reflect changes in the CUD. Errors or unusual conditions in the process are reported to the operator log.

The Consistency Process

The consistency process runs routinely once a day to ensure that the MUD on each Supervisor disk is identical to the NETVAL CUD. The checksum for each CUD block is compared to the checksum for the corresponding MUD block. If the two checksums do not match, data in the CUD block is sent to the Supervisor and written in the MUD block on the Supervisor disk. This event is reported to the operator log.

The automatic consistency process runs on a schedule set by an operator validator. Within the start and stop time defined, NETVAL attempts to run this process simultaneously for all Supervisors. At stop time, the process is automatically halted even if it is still running for a Supervisor.

The automatic consistency process and the MUD update cannot be performed at the same time; therefore, a scheduled consistency process will not begin until all outstanding MUD updates are completed. Once the consistency process is underway, it must finish before the next MUD update can start (however, CUD updates may be performed as usual). The consistency process can also be initiated manually by operator validators.

Updating the Supervisor Class and Group Tables

Origin and destination data is contained in the Class, Group, Host List, and Node List tables on the NETVAL disk. Data is extracted from these four tables and assembled into the Supervisor Class and Group table. In the Supervisor Class and Group table, classes are defined by originating host and node numbers, and groups are defined by destination host numbers. The network administrator validator is authorized to maintain the origin and destination data on these tables. Current Supervisor Class and Group tables are sent sequentially to all network Supervisors and written to each Supervisor disk.

USER CAPABILITIES

NETVAL users validate network usernames. By assigning options and by defining username access profiles, validators also determine network access for other users.

The NETVAL operations that validators can perform are determined by authorization levels and/or control options assigned to validator usernames. A NETVAL user may be a password-only user or may be assigned one of the following three authorization levels: account supervisor, network administrator, or programmer. Control options in the user entry determine whether a validator has account supervisor or network administrator authorization. A password-only user or an account supervisor validator can also be assigned Auditor or Operator control options. Programmer authorization is assigned to one NETVAL user in the network by the inclusion of a username in the NETVAL configuration file. Table 14-2 defines user authorization levels and Table 14-3 defines the Auditor and Operator options.

Table 14-2. NETVAL User Authorization Levels

Level	Capabilities
Password-Only	Change own password only. The password-only user may be assigned the Auditor or Operator options and may not be assigned the No Password option.
Account Supervisor	Display, add, modify, or delete user entry information for usernames in the validators own GAN. The account supervisor validator may be assigned the Auditor or Operator options.
Network Administrator	Display, add, modify, or delete user entry information for any username in the CUD. Validators at this level can set up and maintain the tables on the NETVAL disk. The network administrator validator can also perform all operations allowed by the Operator option.
Programmer	Perform all functions of validators who have account supervisor and network administrator authorization. The programmer validator can also read and write CUD or MUD blocks to compare and correct data.

Table 14-3. NETVAL User Options

Option	Capabilities
Auditor	Display CUD entry information for any username.
Operator	Use the archive subsystem to perform disk and tape operations. Operator users can perform the manual consistency procedure, set the schedule for the automatic consistency process, and halt the automatic consistency process. Operator users can also specify the Supervisors to be accessed for the automatic consistency process, the MUD update, and the Supervisor Class and Group table update. The Operator option allows users to display messages from the operator log.

THE NETVAL DISK MAINTENANCE PROGRAM

The NETVAL Disk Maintenance (NVDM) program is used in conjunction with NETVAL. The NVDM program runs in the same ISIS slot as NETVAL; NVDM and NETVAL cannot run simultaneously.

NVDM is used to initialize or reformat the NETVAL disk and to reconfigure files. NVDM is also used to display information from the NETVAL disk.

The NVDM program has its own archive subsystem, which can be used to perform the tape operations included in the NETVAL archive subsystem. In addition, the NVDM archive subsystem allows the validator to restore files from tape to disk. All tape files or a particular file (such as the CUD file, the MUD Update file, or one of the Node List files) can be copied.

The NETVAL programmer validator and a specially designated NVDM operator validator can use NVDM. Their usernames are contained in the NETVAL configuration file.

RAM

CONTENTS

OVERVIEW

Raw Accounting Merger (RAM) is a program that collects raw accounting data for interpretation and analysis. This accounting information can be used for generating various database reports such as user billing and cost accounting, capacity planning, and network monitoring. RAM assembles this information into session records and writes them on magnetic tape. RAM runs in an Internally Switched Interface System (ISIS) slot in a TYMNET node as shown in Figure 15-1. Figure 15-2 illustrates the various components of RAM.

Each Supervisor writes accounting information to a circular file on a disk. This circular file can store data until previously entered data is overwritten, which may take days or even months depending on Supervisor activity. However, it is important that RAM retrieve the data before it is overwritten.

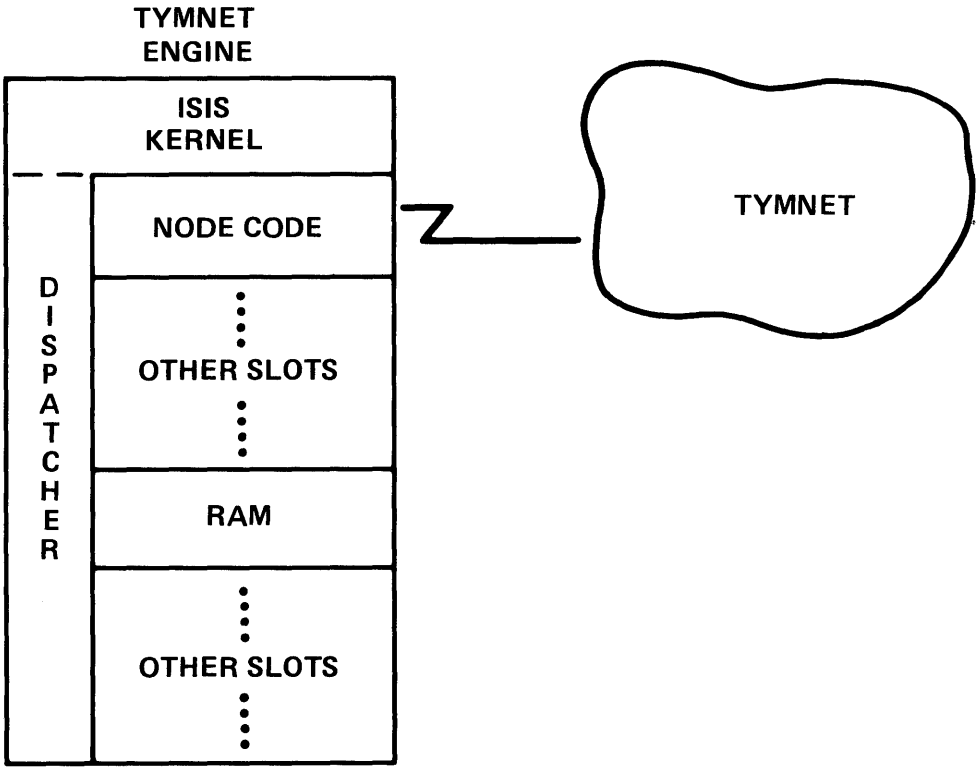


Figure 15-1. RAM in a TYMNET Network

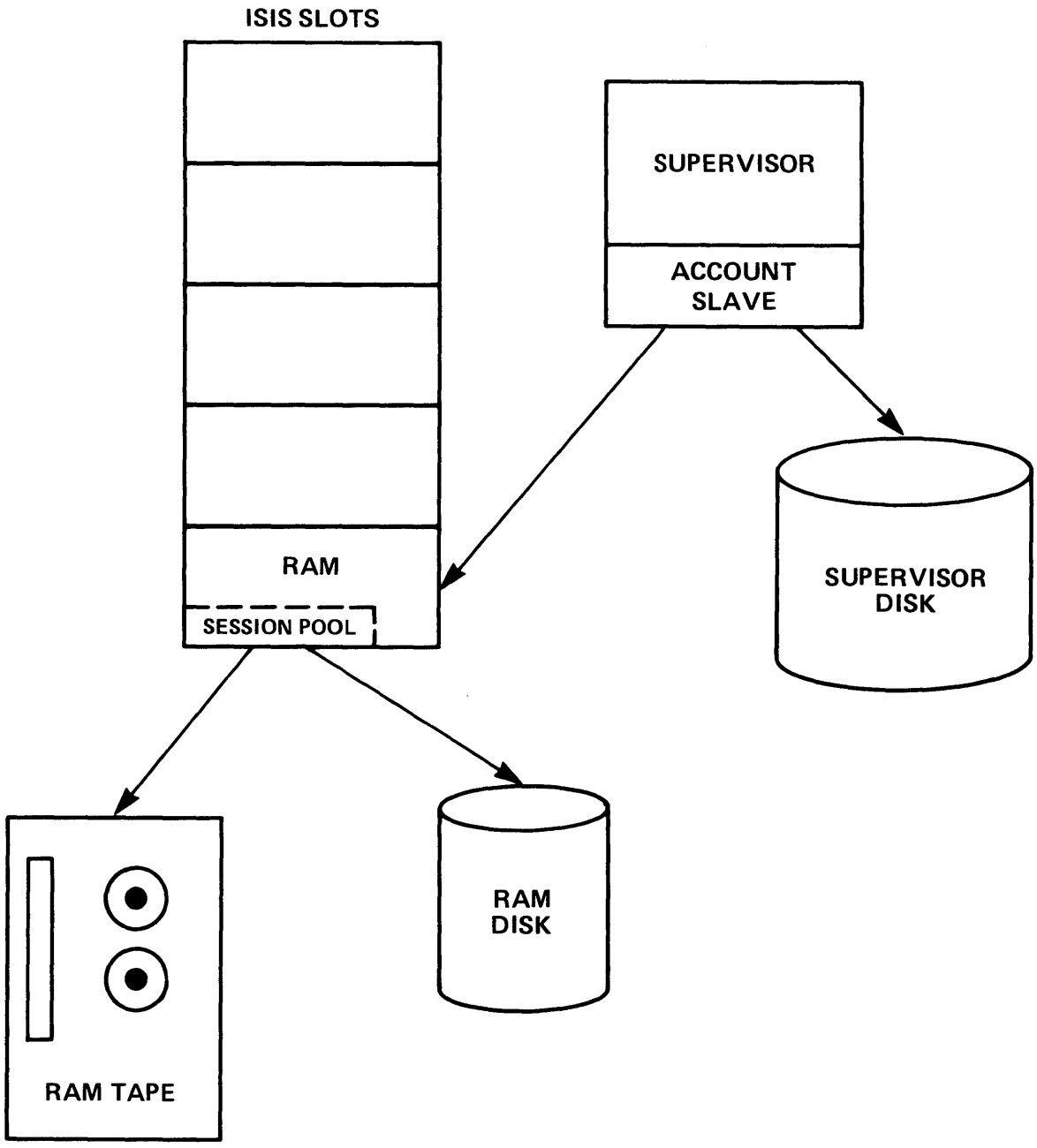


Figure 15-2. RAM Components

To collect accounting data, RAM builds a circuit to each network Supervisor and starts the Accounting slave program. Accounting is a Supervisor slave program with read-only access to the disk file on which the Supervisor writes the accounting information. RAM instructs the Accounting slave program to send it accounting data, one block at a time. Since each network has more than one Supervisor, RAM collects the stored data in chronological order. Figure 15-3 illustrates RAM data collection.

RAM consolidates the accounting information from the Supervisors into session records. A typical session record includes the following:

- session number
- login time
- logout time
- input character count
- output character count
- username
- origination node number
- destination node number

RAM then writes the session records to tape.

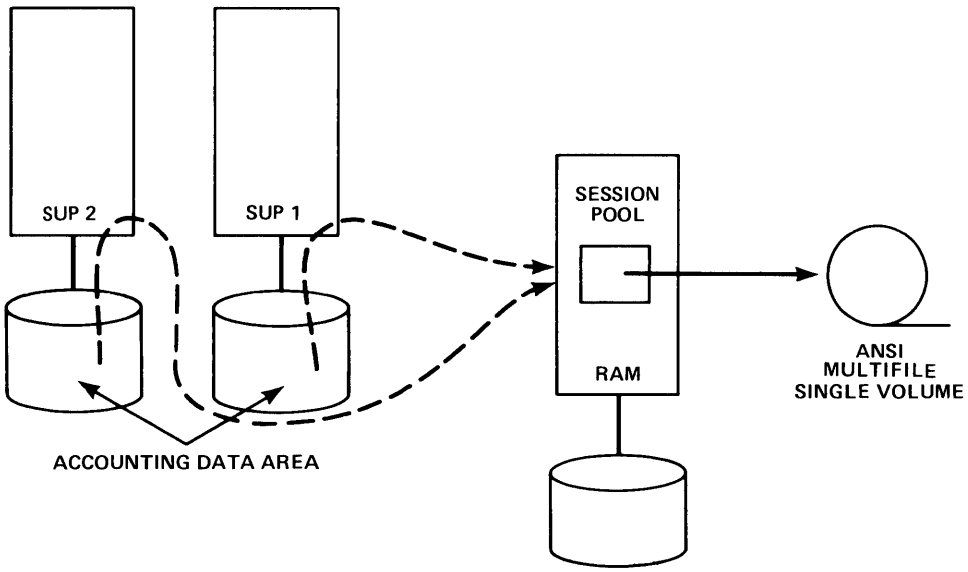


Figure 15-3. Data Collection by RAM

RAM Tapes

RAM may be configured for one or two tape units. The number of tape units is specified at system generation.

Information stored on RAM tapes is formatted to ANSI standards for compatibility with the widest variety of machines on which the information might be processed. During system generation, the user may choose whether the ASCII data stored on tape is written with mark parity or space parity.

RAM Data and the Supervisor

When a user login occurs, the Supervisor generates the initial accounting data for the session. This data, known as the preamble, includes a unique session number that identifies the session. The preamble also includes the following accounting data:

- username
- universal user number
- terminal type
- origination node number
- origination port number
- destination node number
- destination port number

The Supervisor sends the session number to the destination node, a TYMCOM.

Every 15 to 20 minutes, the TYMCOM sends the following information to the Supervisor:

- session number
- updated input character count
- updated output character count

Additional accounting data, such as packet and segment counts, may also be sent when appropriate.

When the session terminates, the TYMCOM sends the Supervisor the final input and output character counts and session termination data. An abnormal termination, for example, a node failure, is treated as if the termination occurred when the Supervisor received the last accounting data.

All accounting data that is created by the Supervisor and all TYMCOM data received by the Supervisor is stored in a circular accounting file on the Supervisor's disk. Accounting data sent from the TYMCOM to the Supervisor is sent in fragments. The Supervisor puts a timestamp on each fragment but does not merge the fragments. When RAM builds a circuit to the Accounting slave program on the Supervisor node, RAM merges this fragmented accounting data into complete session records. RAM then writes the completed records to tape.

RAM Data in Capacity Planning

RAM data can be used for global network projections by looking at such items as network characters transmitted, connect time, and the number of sessions for any given time period (monthly, hourly).

By analyzing such data, network planners can see where network resource usage can be optimized by process scheduling for non-peak hours. For example, the terminal identifier associated with each session record defines a particular set of terminal characters such as baud rate and character set. Within each session record, additional fields provide input/output character counts, as well as connect, start, and stop times. For those sessions to and from X.25, X.28, or X.75 interfaces, additional accounting data is available from RAM, including packet and segment counts, Data Network Identification Codes (DNICs) called, and calling Data Terminal Equipment (DTE).

Traffic statistics can be discerned by looking at the physical input/output port numbers. For example, if a host has 16 available ports and only 8 are being used, that particular host is operating at 50 percent capacity.

ELF

CHAPTER 16

ELF

CONTENTS

OVERVIEW

The Engine Load Facility (ELF) is a program that transfers and loads NAD Image Binary (NIB) code into TYMNET Engine processors and slots. ELF runs in an Internally Switched Interface System (ISIS) job slot in a TYMNET node and builds a circuit from the network through one or more gateways to the TYMNET public network where the code is accessed. In turn, the code is transferred along the circuit to an ELF storage device (disk or tape unit). The binary image of code that is stored on the ELF disk or tape may then be loaded into an entire Engine processor or into an ISIS slot in a node running in the network. One of the advantages of ELF is that once the code has been transferred and stored on an ELF disk or tape, a circuit does not need to be built to the TYMNET public network in order to load the Engine or slot. Also, ELF is used to transfer and load combinations of the following types of code: the Supervisor, Node Code, the kernel, and slot code. ELF may also be used to load diagnostic code.

When an Engine's code is loaded, the target Engine processor is brought up in network. The node can then communicate with the following:

- the Supervisor
- other nodes

When slot code is loaded, the slot host is brought up in the network. One of the slot code programs such as NETVAL, XCOM, or RAM can then be run in an ISIS slot.

Besides loading code, ELF can be used for the following:

- to dump code from an Engine, slot, or partial slot to an ELF disk or tape
- to restart a node that is down, but is loaded with usable code and running the bootstrap program

NETWORK ENVIRONMENT

The ELF environment usually includes at least two networks: the TYMNET public network and a custom network, with at least one gateway between these networks.

An ELF command sequence is used to build a circuit from the custom network through one or more gateways to the TYMCOM-X in the TYMNET public network. If no other networks exist between the custom network and the TYMNET public network, only one gateway is used. However, if one or more networks exist between the custom network and TYMNET, the circuit must be built through multiple gateways. A nonstandard login procedure is available for these conditions.

Code used for loading in the network is initially contained in a NIB file assembled by the Node Assembler/Debugger (NAD) program. NIB files are stored on a TYMCOM-X, which is a PDP-10 mainframe computer that runs Tymshare time-sharing software.

Once the TYMCOM-X is accessed, the NIB file data is transferred back along the circuit through the same gateway(s) to an ELF device. The data is transferred through the network as normal user data traffic.

TRANSFERRING CODE TO AN ELF DEVICE

The NIB file, stored on the TYMCOM-X, may contain Engine code, slot code, or partial slot code (portions of the code necessary for slot execution, i.e., PASCAL object code). In the create process, the circuit built from the ELF node through one or more gateways to the TYMCOM-X allows access of the NIB file. Once the circuit is completed, ELF starts the slave program, ELFSLV. This program resides on the TYMCOM-X and accesses the NIB file, which is then scanned by ELF. ELF accepts records from ELFSLV through the established circuit and keeps track of the transfer of data to its storage device.

Engine code can be transmitted and stored in image format or compressed format. However, slot code and partial slot code are transmitted and stored only in compressed format. In compressed format, the total number of characters is reduced by substituting two characters for certain strings of like characters. However, in image format, data is transmitted and stored byte for byte. Compressed code occupies less storage space and can be saved and loaded faster than image code.

When Engine code is loaded using the ELF program, the code is transmitted and stored in compressed format. However, when Engine code is loaded independently by the bootstrap program on the target Engine processor, the code is stored in image format on an ELF device and transmitted byte for byte to the bootstrap. Note, the decision to specify image or compressed format must be made during the create process.

If compressed format is specified, ELF reports breaks in the address sequence by displaying a series of data addresses and the number of bytes sent to each address. If image format is specified, ELF periodically reports the number of records sent, acknowledged, and lost. ELF indicates when the data transfer is complete. If the code has been transferred to disk, ELF reports the file address, the physical sector address where the file is stored, and the number of disk sectors occupied by the file. The NIB file data is now contained in an ELF file on the ELF device. The Engine code, slot code, or partial slot code in the ELF file is available for loading.

LOADING ENGINE CODE

Once Engine code has been transferred from a NIB file to an ELF device, an Engine may be loaded with either of the following:

- Node Code
- a combination of Node Code, the ISIS kernel, and one or more slot codes bound together by the MERLIN (Merge and Link) program

Loading occurs through a Synchronous I/O board or through the SIO (Serial Input/Output) board on the Engine that is the target Engine processor.

The Conventional Downline Load

Engine code is loaded into a target Engine processor in the network using a conventional downline load process. Prior to loading, the target Engine processor must be down, and its bootstrap program must be running. In addition, because the neighbor node acts as an intermediary between the ELF node and the bootstrap program of the target Engine processor, an accessible network path must exist between the ELF node and the neighbor node.

ELF builds a circuit to the kernel host of the neighbor node; the circuit is continued over the line specified to the bootstrap. Using downline load protocol, ELF sends each record of the ELF file data from the input device to the neighbor node. From the neighbor node, the code is transmitted to the bootstrap program, which sends an acknowledgement back through the neighbor to ELF. The bootstrap loads the code into the target Engine processor's memory.

ELF reports on the data transfer as it occurs. If the file is stored in compressed format, ELF first displays the memory address where the data is being sent. Then ELF periodically displays the number of records sent, acknowledged, and lost. ELF then reports the number of bytes sent to the data address. However, if the file is stored in image format, ELF does not display data addresses and byte counts; ELF displays only the number of records sent, acknowledged, and lost.

When the bootstrap finishes loading the target Engine processor's memory, ELF issues a start-up command. As a result, the node is brought up in the network.

Loading an MXP Node

Besides using the conventional method of loading code, code may be loaded into an ISIS Multiple Extended Processor (MXP) configuration. An MXP node is a cluster of Engine processors. Each of these processors has an MXP number that uniquely identifies that processor in the cluster. (The MXP machine number is not a node number.) The ISIS operating system, with a kernel but no dispatcher, runs on each Engine processor. Each Engine processor has a coprocessor, known as the XPI (Extended Processor Interface), which performs the dispatcher function and is involved in the loading of object code. Each MXP processor also has a Shaman slot. The Shaman provides a network interface to the XPI card on its processor and to the ISIS kernel in other processors in the MXP cluster.

Before using the MXP load option, Node Code must be running on one processor in the MXP node (a conventional downline load from outside the cluster must have been performed to this processor). The other processors in the cluster, which are not running code, are then loaded using the MXP option as follows:

A circuit is built from ELF to the Shaman host in the previously loaded processor in the MXP cluster. The code is transmitted through a series of communication links along the following path:

1. from the Shaman host to the XPI in the previously loaded processor
2. from this XPI to the XPI in the target processor
3. from the target XPI to the target processor bootstrap program

The bootstrap program then automatically loads the code into the target processor's memory. ELF reports on the data transfer by displaying the address where the data is being sent; by periodically displaying the number of records sent, acknowledged, and lost; and by displaying the number of bytes sent.

LOADING SLOT CODE AND PARTIAL SLOT CODE

Once slot code or partial slot code has been transferred from a NIB file to an ELF device, the code can be loaded at any time. Slot code, which runs in an ISIS slot, exists for such programs as ELF, NETVAL, XCOM, and RAM. Partial slot code, which runs on part of an ISIS slot, is usually object code from the Concurrent PASCAL Compiler.

Loading Slot Code

When slot code is loaded, the node that is identified by the kernel host must be up and running in the network. ELF builds a circuit from its node to the kernel host of the node in which the ISIS slot will be loaded. (The ISIS slot may be located in the ELF node or in another node in the same network.) ELF communicates with ISIS through the circuit to the kernel host.

ELF uses ISIS DDT (Dynamic Debugging Tool) protocol to direct ISIS to initialize the slot and to write the slot code that will be transmitted to memory. Segment 0E of the slot code is transferred first and loaded into the beginning of the slot's memory, enabling the Memory Access Controller (MAC) to be initialized. The data in segment 0E prepares ISIS for the loading of subsequent file data in the slot.

After displaying the segment 0E data address and the number of bytes sent to that address, ELF periodically reports on the transfer of the remaining slot code. ELF displays a series of data addresses and the number of bytes sent to each address.

ELF continues sending the slot code in ISIS DDT protocol until all the data has been transmitted. ELF sends the start-up command for the slot, which is acknowledged by ISIS. After the slot begins to run the code, the slot host is brought up in the network.

Loading Partial Slot Code

Partial slot code loading is similar to slot code loading. However, the main difference is that the MAC needs to be initialized prior to the beginning of the loading process. When partial slot code is loaded, segment 0E of the slot code is not transmitted, and ISIS is not directed to initialize the MAC. Otherwise, the loading process is the same for both types of code.

The node that is identified by the kernel host must be up and running. The partial slot that is loaded may be in the ELF node or in another node in the network. ELF performs the following functions during the loading of partial slot code:

1. Builds a circuit from its node to the kernel host of the target node to communicate with ISIS.

2. Uses ISIS DDT protocol to write the partial slot code to memory.

3. Reports on the data transfer by displaying a series of data addresses and the number of bytes sent to each address.

4. Sends the start-up command for the slot. After the slot begins to run the code, the slot host is brought up in the network.

DUMPING CODE

The dump process is the reverse of the load process. Instead of code being copied to an Engine processor or to a slot, as in the load process, code is copied from an Engine processor or from a slot to an ELF device.

Dumping Engine Code

When Engine code is dumped, the code is read from the memory of a target Engine processor and written to an ELF disk or tape file. The target Engine processor cannot be running in the network; however, the bootstrap program on the target Engine processor must be running. A neighbor node is used as an intermediary between the bootstrap program on the target Engine processor and the ELF device.

ELF builds a circuit to the kernel host of the neighbor node and uses downline protocol to request the Engine code from the neighbor node. The neighbor node communicates, in turn, with the bootstrap program of the target Engine processor. Each record is sent from the bootstrap through the neighbor node and is written in image format to the ELF disk or tape. ELF reports on this data transfer by displaying the number of records requested from the target Engine processor, the number received by ELF, and the number of records lost. The dump operation is complete when all the requested Engine code is transferred and stored on the ELF device.

Dumping and reloading Engine processors provides valuable information about their state and allows the Engine processors to be returned to service. (An Engine processor that is down but loaded with usable code can also be returned to service by using the restart process.) Data acquired from a dump can help technical support personnel analyze code problems offline.

Dumping Slot Code and Partial Slot Code

When slot code is dumped, the code is read from an ISIS slot and written to an ELF disk or tape file. ELF builds a circuit to the kernel host of the node from which the slot code will be dumped. The slot code is transmitted along this circuit to the ELF device. Segment 0E of the code is transferred first. This segment contains information on the subsequent code to be dumped from the slot.

ELF reports on the data transfer by periodically displaying the addresses from which the code is copied and the number of bytes transferred from each address. The dump operation is complete when all the code in the slot has been transferred to the ELF device.

Dumping partial slot code is similar to dumping slot code, except for a few differences. Since only part of the code in the slot is transferred, the segments to be dumped must be specified. ELF builds a circuit to the kernel host of the node where the ISIS slot is running. The partial slot code is transferred to the ELF device. ELF reports on the data transfer by displaying the addresses and the number of bytes transferred from each segment. The dump operation is complete when the specified segments of code have been dumped to the ELF disk or tape.

RESTARTING A NODE

The restart procedure is used when the target Engine processor is down, is loaded with usable code, and is running its bootstrap program. This operation is performed through a node that is a neighbor to the target Engine processor.

The restart procedure may be used when a start address does not exist for code that was previously loaded into an Engine. This procedure can also be used to restart code that was previously running in the target Engine processor.

1. ELF builds a circuit to the kernel host of the neighbor node; the circuit is continued to the bootstrap program of the target Engine processor.

2. The address where Engine code should be started must be specified.

3. ELF transmits this start address to the target Engine processor and receives an acknowledgement that the data was received.

4. The target Engine processor starts up and is now running in the network.

ELF DEVICES

In order to use the ELF program, one or more ELF storage devices (disk or tape) must be available on the node where the ELF program runs. Code from NIB files is transferred to an ELF device where it is stored prior to the load process. Code is dumped from Engine processors and ISIS slots to an ELF device.

A specified file or all files can be copied from one ELF device to another. All ELF devices can be searched for a specified file, and information on that file can be displayed.

Each ELF disk or tape unit is identified by a device unit identifier. For a disk, this identifier is the character "d" followed by a decimal number in the range 0 through 7. For a magnetic tape, the device unit ID is the character "m" followed by a decimal number in the range 0 through 3. (These ranges may not be as great in your ELF configuration.)

Prior to the transfer of code from a NIB file to an ELF device, the disk or tape unit must be ready to receive the data. ELF commands are used to reallocate ELF disk space and to properly position an ELF tape for receiving NIB file data.

ELF Disk

An ELF command is available to determine the total amount of space that exists on a disk. Disk space can be reallocated in different ways to accommodate code transferred during a create, dump, or copy process. Files can be repositioned to create more contiguous disk space and/or deleted from a disk.

The following disk parameters and information can be displayed:

- ISIS logical unit number
- physical sector address
- number of sectors on the disk
- file address
- file type
- file identifier
- version of ELF used to create an ELF file
- NIB file creation date
- ELF file creation date

A disk initialization procedure is used to erase all disk files and format the disk for use by ELF. This process is completed when the ELF program is first brought up on a host, when a disk needs to be replaced, or when a disk has been destroyed.

ELF Tape

Commands are available to properly position an ELF tape before a write operation (in the create, dump, or copy process) or before a read operation (in the load or copy process). An ELF tape can be returned to its beginning-of-tape mark. Also, the tape can be moved forward to skip over existing files. Either all files or a specified number of files can be skipped. In addition, parameters and file information can be displayed for an ELF tape unit.

Configuration Management Facility

To be supplied at a later date

ONTYME

To be supplied at a later date

Glossary

To be supplied at a later date

READER COMMENT FORM

TYMNET Network Technology Development/Documentation
10261 Bubb Road
Cupertino, CA 95014

This document has been prepared for use with TYMNET products. Should you find any errors or problems in the manual or have any suggestions for improvement, please return this form to the TYMNET NTD Documentation Department. Include page numbers or section numbers, where applicable.

Product Name _____

Version Number _____ Document Date _____

Document Title _____

Does the manual meet your needs? Yes No

Do you find the material:

- Easy to read and understand? Yes No
- Organized for convenient use? Yes No
- Written for your technical level? Yes No

Comments _____

_____

_____

_____

_____

Your Name _____ What is your occupation? _____

Company Name _____

Address _____

_____

Please return this form to: Manager, Documentation
TYMNET
Network Technology Development
10261 Bubb Road, Building D
Cupertino, CA 95014

TYMNET

Network Technology Development

10261 Bubb Road
Cupertino, CA 95014
408/446-6632

Publication Number 57

January 15, 1985